JORG WALKOWIAK

MICRO APPLICATION

8

AMSTRAD

GRAPHISMES ET SONS DU CPC 464



UN LIVRE DATA BECKER

JORG WALKOWIAK

MICRO APPLICATION

8

AMSTRAD

GRAPHISMES ET SONS DU CPC 464



UN LIVRE DATA BECKER

Distribué par MICRO APPLICATION
147 Av. Paul Doumer
92500 RUEIL-MALMAISON

et également

EDITIONS RADIO
3 rue de l'Eperon
75006 PARIS

ISBN 2-86899-016-9

Traduction Française et mise en pages assurées par Roland KANNENGIESER

Edité par Frédérique BEAUDONNET
Léo BRITAN
Philippe OLIVIER

REPERTOIRE

Préface

POSSIBILITÉS GRAPHIQUES ET SONORES

Les techniques d'avant-garde ont mis à la portée des ordinateurs familiaux les plus modestes des capacités graphiques et sonores auparavant domaines réservés aux systèmes informatiques spécialisés et onéreux.

Mais où est l'intérêt de savoir que son ordinateur est équipé d'un microprocesseur graphique comme le 6845-CRTC de COMMODORE ou d'un générateur de sons multicanaux comme le AY-3-8912 de GENERAL INSTRUMENTS si l'on ne sait pas en exploiter les caractéristiques pour obtenir la représentation graphique d'une fonction mathématique ou encore utiliser son clavier comme un orgue.

Bien entendu, le manuel d'utilisation fourni avec l'appareil indique comment utiliser les différentes instructions disponibles pour adresser ces circuits spécialisés; mais l'algorythme, c'est à dire la solution à un problème donné, sera à définir par l'utilisateur, ou à prendre dans une littérature adéquate.

Ce livre doit vous épargner ceci, il doit être le conseil indispensable pour vous aider à résoudre toutes questions de programmation graphique ou sonore.

La première partie vous fera passer de la création du point à l'image tridimensionnelle à l'aide de nombreux exemples de programmes vous démontrant en théorie et pratique les techniques et capacités de la programmation graphique.

Il en va de même pour la programmation de sons, bruits et mélodies qui est traitée dans le dernier chapitre du présent livre.

Après avoir lu et assimilé cet ouvrage, vous n'aurez pas seulement de meilleures connaissances, mais disposerez également d'une bibliothèque

de programmes représentant déjà bien plus que le prix de ce livre si vous les achetiez sur cassette ou disquette.

Les programmes de représentation graphique de fonctions en deux ou trois dimensions seront particulièrement appréciés des étudiants.

Les personnes ayant le goût du dessin auront la plus grande joie d'utiliser l'éditeur graphique et en feront leur nouvel appareil à dessiner.
En particulier les programmeurs de jeux d'arcade et d'aventure auront la tâche facilitée; ce sera un vrai jeu d'enfant que d'écrire les routines necéssaires à partir de ce programme.

Quant à l'homme d'affaires, il ne se lassera pas d'utiliser les programmes permettant de visualiser les statistiques quelles qu'elles soient, et sous diverses formes.

Bien entendu, pas question de s'ennuyer; c'est pourquoi les fervents amateurs de jeux trouveront tout au long de ce livre des trucs et astuces concernant la programmation de jeux d'action, ainsi qu'un jeu complet également décrit.

Mais tout le monde n'est pas forcément un as du tir ou du pilotage de vaisseau intergalactique, celui qui préfère la musique se penchera donc plutôt sur notre mini orgue.

Quoiqu'il en soit, je pense avoir pu réveiller votre intérêt à l'aide de ces quelques exemples; je puis donc arrêter ici cette préface et attaquer le sujet principal!

Et en tout cas, je suis sûr que les essais que vous ferez à l'aide de ce livre vous procureront autant de plaisir que nous en avons éprouvé lors de la mise au point des divers programmes contenus dans cet ouvrage.

En dernier lieu, je tiens à remercier de tout coeur Mme. Alicia Clees et Mr. Claus Wagner pour leur précieuse aide et collaboration qui ont permis de réaliser cet ouvrage.

Recklinghausen,
Mars 1985 Joerg Walkowisk

Chapitre 1

INTRODUCTION

L'évolution de la communication

Si l'animateur d'un jeu demande à ses candidats 'quelle est la langue internationale ?' ou 'quelle est la langue la plus courante dans le monde ?', il lui serait probablement répondu 'la langue anglaise naturellement!'.

Qu'en serait-il si la réponse était 'l'image!'?

Comment les hommes s'exprimaient-ils il y a plusieurs millénaires, alors qu'ils se déplaçaient à quatre pattes dans un paysage hostile?

Se contentaient-ils de leur vocabulaire ou utilisaient-ils un langage de gestes et donc d'images?
Peut-être traçaient-ils des traits dans le sable, créant une image, une carte indiquant aux autres où se trouvaient de bons terrains de chasse?

Qu'y avait-il avant les écrits et les fresques murales - de simples dessins au trait fait de cendre portée sur les parois de cavernes et à l'aide desquelles, aujourd'hui encore, les historiens peuvent suivre la vie de tribus complètes?

Quels étaient les moyens de communication plusieurs millénaires après?

Regardons les hiéroglyphes de l'Egypte ancienne. Les dessins sont devenus une écriture; mais chaque lettre est encore une image.

Passons encore quelques millénaires pour arriver à notre époque.
Notre langage évolué nous suffit-il ou utilisons nous encore des symboles, des images?

Combien d'européens connaissent-ils le sens du mot français 'paix'; mettez en rapport le nombre de personnes pour qui la colombe blanche a une signification!
De même, dans le monde, combien de personnes connaissent l'image d'un

coeur transpercé d'une flèche et combien connaissent le petit couple de bande dessinée 'L'amour, c'est...'?

Les images sont donc toujours présentes de nos jours; elles ont le mérite d'être comprises en un clin d'oeil.

Nous en sommes tous conscients; comment expliquer autrement l'importance qu'ont pris photographie et cinéma?

Ce n'est pas par hasard que nous trouvons des revues comprenant des photographies en quadrichromie, les informations télévisées, les bandes dessinées qui sont certainement la seule lecture qui interèsse les plus jeunes grâce aux dessins très expressifs, pleins de sentiments, mais n'oublions pas les revues techniques qui sont elles aussi remplies de dessins, plans et éclatés.

Tous sont d'accord: 'un dessin parle plus qu'un discours!'. Il n'est donc pas étonnant si nous y faisons ici et là appel.

Ceux-ci ont en outre l'avantage de pouvoir être adaptés aux besoins:
Les photos de vacances doivent être en couleur et pleines de détails pour rendre l'ambiance lors de la prise de vue et laisser la meilleure impression lorsque vous les montrez.

Un agent immobilier montrera à l'acheteur potentiel d'une maison tout d'abord un poster afin qu'une première décision puisse être prise. Par contre, un architecte devant effectuer des travaux n'aura aucun besoin d'une photo; il utilisera plutôt une série de plans reflétant les diverses vues intéressantes, de même qu'un architecte d'intérieur ne pourra se passer de plan.

L'un se contente donc d'un plan ou dessin ne reprenant que les principales informations alors que l'autre, l'acheteur, attachera également de l'importance à des détails apportant plus d'informations, par exemple sur les couleurs, l'éclairage, etc... pour se faire une

meilleure image.

Mais les schémas ne sont pas seulement utiles quand il s'agit de refléter les caractéristiques physiques d'un objet, mais aussi dans le cas de la représentation graphique de chiffres.

Aux statistiques les courbes de Gauss, à l'étudiant la représentation de fonctions et à l'homme d'affaires les 'camemberts' et autres graphiques représentant les chiffres d'affaires des derniers mois.

Egalement dans la vie courante, nous sommes toujours plus confrontés à des images synthétiques ne reprenant que l'essentiel.

Ce sera une station de télévision qui sera fière de son nouveau sigle et pensera que ses programmes se seront améliorés seulement avec ce sigle, cette représentation de l'image de marque.

Ce sera le dessinateur industriel qui développait ses produits sur la planche à dessin durant de nombreuses heures à l'aide de crayons, règles et calculatrice. Maintenant, ce travail fastidieux sera réalisé en un tour de main avec des moyens électroniques sur un écran.

Tous ces utilisateurs profitent du développement très rapide des techniques informatiques qui permet maintenant aux PME, PMI et même aux particuliers d'acheter et utiliser des systèmes graphiques informatiques.

Tous contribuent à l'évolution de notre langage qui s'enrichit d'un nouvel aspect, le dessin informatique.

Car depuis longtemps, la concurrence bat son plein que ce soit pour trouver les meilleurs dessins d'ordinateurs, pour trouver des clients faisant appel à d'importants systèmes informatiques graphiques dans leurs campagnes publicitaires. Et même le premier film réalisé uniquement à partir d'images synthétiques est à nos portes à l'heure

où nous mettons ce livre sous presse.

LE GRAPHISME SUR ORDINATEUR

C'est là un sujet d'actualité.
Que ce soit un mini ordinateur ou un système informatique important, il n'est plus question pour l'utilisateur d'utiliser un ordinateur ne pouvant que calculer et faire du traitement de texte.
Ne nous étonnons donc pas si les arguments avancés par les différents constructeurs sont maintenant une plus grosse capacité mémoire, un accès plus rapide à celle-ci, une définition écran plus fine et encore plus de couleurs. Où en est l'aspect ergonomique du clavier qui était l'argument de vente il y a peu de temps?

Quelles sont les possibilités de cette nouvelle technologie?

Dans le secteur professionnel, ces capacités graphiques sont présentes dans de nombreux domaines; dans beaucoup d'entre eux, cette évolution n'a été possible que grâce aux progrès techniques réalisés.

A l'aide de programmes adéquats, un calculateur puissant remplace tout un avion - tous les aspects d'un vol ainsi que leurs représentations peuvent être réalisées avec réalité sur un simulateur de vol.

En fait, une formation sur simulateur de vol est maintenant mise au même niveau qu'une formation en vol réel. C'est aussi la raison pour laquelle les grandes compagnies aériennes reconnaissent et utilisent ce type de formation, ce qui réduit en outre considérablement le coût.

Impensable de se passer d'un ordinateur dans le domaine des sciences physiques; personne ne pourrait exploiter rapidement la foule d'informations dont il faut tenir compte. De même, aucun professeur n'a en tête toutes les connaissances humaines; l'ordinateur est donc devenu indispensable dans les secteurs tels que la physique des plasmas ou la génétique qui font appel à des relations complexes.

Depuis longtemps, les capacités mentales ne sont plus suffisantes à

elles seules pour faire avancer les chercheurs dans leurs travaux. Si jusqu'à ce jour, un savant pouvait se représenter dans son imagination le résultat de ses calculs, c'est devenu tout à fait impossible quand il faut comparer en génétique les structures héréditaires ou encore certaines molécules.
Un terminal graphique représentant sous tous ses angles une molécule, en permettant tous agrandissements et réductions est devenu une aide indispensable.

Bien entendu, il n'y a pas que les sciences atomiques qui utilisent de tels systèmes, ils ont également leur place dans le développement et la réalisation d'automobiles, de maisons et même d'immeubles.
Et si une table traçante est reliée à l'unité de calcul, un architecte pourra donner libre cours à ses élans créateurs.
Les travaux de routine, les calculs, le dessin à l'échelle de toute les pièces sous tous les angles, etc... pourront être effectués durant une pause ou la nuit.

Avec l'aide de l'ordinateur, d'où le terme de DAO (Dessin Assisté par Ordinateur), un architecte peut réaliser en un tour de main des travaux qui auraient demandé plusieurs semaines.
Dans la fabrication, les données calculées peuvent être mises en mémoire dans des machines appropriées (tour CNC, centre d'usinage CNC...), permettant la réalisation matérielle et entièrement automatique d'un objet créé sur ordinateur.
Ce procédé s'appelle la CFAO (Conception et Fabrication Assistée par Ordinateur).

Ces capacités graphiques ne sont pas seulement mises à profit dans la recherche et l'industrie, mais de plus en plus dans des domaines artistiques.

Au Japon et en Amérique, des studios de création graphique par

ordinateur voient le jour et nous montrent qu'il est possible de montrer à l'écran les objets les plus singuliers en toute réalité. La qualité graphique, en ce qui concerne le piqué de l'image et le rendu des couleurs, est tellement bonne que le premier film de science fiction est en cours de réalisation au 'New-York Institut of Technology'. Il devrait sortir sur les écrans sous peu - son titre 'The Works'.

Si les studios Walt Disney ont encore mixé des séquences d'images réelles et synthétiques dans 'TRON', les producteurs New Yorkais ne font plus appel qu'à des images synthétiques.
Il semblerait que le cinéma conventionnel ait fait son temps.
Dans quelques années, les acteurs bien payés auront peut être peur de leurs 'collègues' les ordinateurs!

Cet exemple montre bien la qualité extrême du graphisme sur ordinateur; les plus petites nuances peuvent être reproduites à l'écran.
Ceci permet les applications médicales - d'infimes modifications des tissus cellulaires peuvent être visualisées à l'écran. Le médecin peut prendre les mesures qui s'imposent et aider de nombreuses personnes.

Une présence dans tous ces secteurs serait impensable si ces images ne pouvaient être représentées dans le détail, avec toute la palette de couleurs nécessaire, tout ceci avec les moyens techniques correspondant.
Ces exemples montrent bien entendu que les caractéristiques graphiques d'un système doivent être adaptées à l'utilisation envisagée.
Le développement est constant, de nouveaux chemins sont toujours explorés; un constructeur copie un système existant tout en l'améliorant alors qu'un autre choisira une toute autre voie et présentera un système inédit.

Heureusement, ce ne sont pas que les magnats de la finance qui en profitent, les retombées sont perceptibles pour le simple particulier.

Car le plus petit utilisateur, et en particulier le plus jeune, attachera de l'importance aux points, lignes, 'sprites' et 'shapes', en un mot aux formes et couleurs.
Le graphisme électronique n'est donc pas seulement un leitmotiv de l'industrie, mais c'est aussi un mot clé dans le coeur de nombreux enfants et adolescents qui attachent beaucoup d'importance à avoir chez-eux les mêmes jeux que dans les salles de jeux.

Impossible de vendre un ordinateur qui ne permette pas de rendre à l'écran un paysage interstellaire en couleur. Il n'est donc pas étonnant que les constructeurs représentés en informatique domestique donnent à leurs ordinateurs des possibilités graphiques de base à l'aide desquelles l'on obtient des résultats tout à fait satisfaisants et respectables.

Chapitre 2

BASES

Avant d'entrer dans le détail du sujet et d'aborder les problèmes tels que la rotation de dessins, la représentation d'échelles graduées ou d'images 'miroirs', nous passerons en revue les instructions graphiques de base de l'AMSTRAD CPC à l'aide de cours exemples.
Les lignes suivantes vous sembleront très proches de celles du manuel d'utilisation de votre CPC. Malgré ceci, ne passez pas ce chapitre sans le lire; vous pourrez toujours en tirer quelque renseignement complémentaire important par la suite

Les listings de programmes

sont imprimés dans cet ouvrage sous la même forme qu'ils apparaissent à l'écran de votre ordinateur quand celui-ci est en mode standard.
Ceci permet un contrôle rapide du programme au cas où il ne fonctionnerait pas correctement.

Certaines routines que nous présentons seront utilisées dans des programmes différents.
C'est pour cette raison qu'ils ont en partie été écrits en tant que sous-programme. Evitez donc de renuméroter les lignes à l'aide de l'instruction BASIC 'RENUMBER'.
Il vous sera ainsi possible de prendre l'instruction 'MERGE' pour insérer ces routines dans vos programmes et vous en épargner l'entrée au clavier.

La technologie

à laquelle fait appel notre CPC 464 ne nous permet pas de réaliser du dessin animé, mais nous mène tout de même à des applications spectaculaires et utiles.

Comme nous l'avons vu précedemment, un bon graphisme recquiert une bonne définition et une large palette de couleurs.

Bien entendu, notre ordinateur qui coute entre 3000 et 4000 ff est

loin de la puissance d'un CRAY de plusieurs millions de francs et qui peut afficher sur un terminal graphique un million de points; mais nous pouvons tout de même afficher 128.000 points - qui se nomment dans le jargon informatique PIXEL -. Dans une liste comparative entre les différents ordinateurs courants comme les Apple, Atari, Sinclair et Commodore, notre AMSTRAD CPC est en première place.

Vous comprendrez tous que l'ordinateur doit se mettre en mémoire si un point doit être allumé ou éteint; si ceci se double encore d'une couleur bien définie, la place mémoire occupée sera encore plus importante.

Pour cette raison, les constructeurs doivent trouver des compromis qui peuvent être les suivants:

1. haute définition, peu de couleurs
2. basse définition, beaucoup de couleurs
3. haute définition, couleurs différentielles

Les deux premiers points ont été retenus sur notre ordinateur AMSTRAD; la troisième solution serait réalisable sur un ordinateur familial, mais le prix des composants, ainsi que du moniteur qui devrait être de meilleure qualité, feraient tellement monter le prix de vente que cet appareil ne serait plus accessible.

Tout au moins à ce jour, puisqu'en 1977, les ordinateurs disposaient au mieux de 16 KO et d'une résolution de 127 x 48 points (TRS 80); aujourd'hui, cette place mémoire a quadruplé en passant à 64 KO permettant une résolution de 600 x 200 points (AMSTRAD CPC).
Les communiqués de presse de certains fabricants de composants annoncent les premiers prototypes de puces d'une capacité d'un méga-octet. Quand ceux-ci entreront en fabrication de grande série et ne couteront plus que quelques milliers de francs, nous assisterons encore à un bond en avant dans les capacités de nos ordinateurs

familiaux.

Les outils

Chacun d'entre nous a déjà essayé une activité artistique, que ce soit au jardin d'enfants avec de la peinture à l'eau, à l'école avec un crayon à papier et de l'encre de Chine ou encore à la maison avec de la peinture à l'huile.

Tous ces exemples ont un point commun: tout comme le peintre installe d'abord son chevalet, nous avons pris de quoi dessiner, avons choisi un support ainsi que notre couleur préférée. Et c'est seulement après mûre réflexion que nous avons vraiment commencé à dessiner.

Quand nous utiliserons notre CPC, nous choisirons d'abord le fond, le papier et la couleur souhaitée.

PAPER

Même ceux qui ne comprennent pas l'Anglais savent de quoi il s'agit. Cette instruction permet de choisir la couleur du fond parmi les 26 ci-dessous.

```
10 MODE 0:PAPER 0:PEN 1
20 FOR FARBE=0 TO 26
30 INK 0,FARBE
40 INK 1,FARBE+1
50 LOCATE 1,1:PRINT "FARBE: ";FARBE
60 FOR I=1 TO 1000:NEXT I
70 NEXT FARBE
```

Ce petit programme fait passer à l'écran les 27 différentes couleurs possibles:

0	noir	1	bleu
2	bleu vif	3	rouge
4	magenta	5	mauve
6	rouge vif	7	pourpre
8	magenta vif	9	vert

10	turquoise	11	bleu ciel
12	jaune	13	blanc
14	bleu pastel	15	orange
16	rose	17	magenta pastel
18	vert vif	19	vert marin
20	turquoise vif	21	vert citron
22	vert pastel	23	turquoise pastel
24	jaune vif	25	jaune pastel
26	blanc brillant		

Complétez maintenant le programme précédent avec les lignes ci-dessous. Vous pourrez revoir toutes les couleurs tranquillement et selon vos goûts.

```
90 INPUT "COULEUR DE FOND";COULEUR
100 INK 0,COULEUR:INK 1,COULEUR+1
110 GOTO 90
```

BORDER

Vous aurez certainement remarqué qu'il reste toujours autour de l'écran un cadre sur lequel vous ne pouvez écrire.
Si vous n'aimez pas cete bordure, faites la disparaitre à l'aide de l'instruction 'BORDER c', où 'c' est un chiffre de 0 à 26 correspondant à la même couleur que le fond.

PEN, INK

'PEN' sera la couleur de l'écriture. Si vous n'avez pas encore essayé cette instruction lors de la lecture du mode d'emploi, imaginez-vous un stylo dans son étui.
Tout comme il y a des stylos de plusieurs couleurs dans un étui, le CPC dispose également de plusieurs couleurs pour son écriture. Le nombre de couleurs dépendra du mode écran sélectionné.

Chaque 'stylo' correspond donc à une couleur, le stylo 0 à une mine de couleur 1 (bleu) et le stylo 1, à une de couleur jaune (24).
Les valeurs standard de chaque stylo dépendent du mode écran sélectionné.
Le mode 2 par exemple ne permet d'utiliser que deux couleurs; les autres stylos auront donc alternativement les mêmes couleurs.

En mode écran 1, l'AMSTRAD peut afficher simultanément 4 couleurs; les groupes de couleurs bleu, jaune, bleu-vert et rouge se répètent donc constamment.

Les stylos 14 et 15 ont une propriété intéressante en mode écran 0; ils écrivent tous deux alternativement en deux couleurs.

Si le choix de couleurs ne vous plait pas, tout comme vous pouvez changer la mine d'un stylo, vous pouvez également attribuer d'autres couleurs à votre 'stylo', en utilisant l'instruction INK. Les codes couleurs à utiliser sont ceux mentionnés dans le tableau ci-dessus. Si vous attribuez deux couleurs à un stylo, vous obtiendrez encore un jeux de couleurs déroutant.

Ces couleurs peuvent également être données au fond et au cadre, qui correspondent eux aussi à un stylo.

Le petit programme ci-dessous schématise bien l'utilisation des instructions PEN et INK:

```
10 REM barres de couleur
20 MODE 0:PAPER 0
30 ligne$=STRING$(40,143)
40 FOR x=1 TO 13
50 PEN x
60 PRINT ligne$
```

```
70 NEXT x
80 FOR couleur=0 TO 26
90 coul2=couleur
100 FOR x=1 TO 13
110 INK x,coul2
120 coul2=coul2+1
130 IF coul2>26 THEN coul2=0
140 FOR i=0 TO 1  :NEXTi
150 NEXT x
160 NEXT couleur
170 GOTO 30
```

<u>Le mode écran</u>

Vous le choisirez en fonction de vos besoins. Les caractéristiques en sont:

1. le nombre de couleurs
2. la taille des caractères
3. la définition

En mode 0 vous disposez en simultané de seize des 27 couleurs. Vous avez donc seize stylos, PEN.
Chaque caractère est assez grand; vous n'avez que 20 caractères par ligne.
De même, chaque point est assez gros puisque la définition n'est que de 160 * 200 points.
Ce mode écran est tout à fait adapté à la création de titres; vous n'aurez probablement pas l'occasion de programmer dans ce mode de fonctionnement.

En mode 1, vous avez le réglage standard, 40 caractères sur 24 lignes, quatre stylos et une définition de 320*200 points.

Ce mode de fonctionnement n'est pas seulement considéré comme mode standard parce que le CPC se trouve dans ce mode à la mise sous tension, mais aussi parce qu'il s'agit là des données d'écran typiques d'un ordinateur familial.
Ceux-ci ne sont jusqu'àlors en aucun cas fournis avec un moniteur, mais doivent être branchés sur le téléviseur qui ne peut techniquement pas afficher un nombre supérieur de points dans de bonnes conditions.

C'est également la raison pour laquelle le CPC 464 peut être définit comme ordinateur famillial couleur; la définition graphique de 640 * 200 points en mode 2 surpasse les caractéristiques d'un IBM PC!
Il en est de même de l'affichage sur 80 colonnes que l'on ne retrouve d'habitude que sur des ordinateurs professionnels.

Le nombre de couleurs ainsi que la dimension des caractères sont maintenant des critères de sélection suffisants.
La définition n'est naturellement pas aussi simple à apprécier.
Au mieux, cette différence peut se faire à l'aide du programme suivant, qui vous montre le pouvoir de résolution à l'aide d'une mire, tout comme le font aussi les émetteurs TV.

```
10 REM demonstration: definition
20 REM ---------------------------------
30 PAPER 0:INK 0,12:BORDER 12:PEN 1:INK 1,0
40 y1=100:y2=300
50 CLS:INPUT "mode de fonctionnement (0,1,2)";MODus:
  IF (MODus>2 OR MODus <0) THEN GOTO 50 ELSE MODE MODus
70 LOCATE 1,1:PRINT "mire de resolution"
80 PLOT 80,100:DRAW 550,100:DRAW 550,300:DRAW 80,300:DRAW 80,100
90 TAG
100 FOR x=80 TO 290 STEP 21
110 GOSUB 390
120 NEXT x
130 MOVE 76,98:PRINT CHR$(240);"20";
140 FOR x=290 TO 400 STEP 11
150 GOSUB 390
160 NEXT
170 MOVE 296,98:PRINT CHR$(240);"10";
180 FOR x=400 TO 450 STEP 5
190 GOSUB 390
200 NEXT
210 MOVE 396,98:PRINT CHR$(240);"5";
220 FOR x=450 TO 490 STEP 4
230 GOSUB 390
240 NEXT
250 IF MODus>0 THEN MOVE 446,98:PRINT CHR$(240);"3";
260 FOR x=490 TO 520 STEP 3
270 GOSUB 390
280 NEXT
290 IF MODus>0 THEN MOVE 476,98:PRINT CHR$(240);"2";
300 FOR x=520 TO 540 STEP 2
310 GOSUB 390
320 NEXT
330 IF MODus>1 THEN MOVE 496,98:PRINT CHR$(240);"1";
340 FOR x=540 TO 550
350 GOSUB 390
360 NEXT
370 TAGOFF
380 IF INKEY$="" THEN 380 ELSE 50
390 PLOT x,y1:DRAW x,y2:RETURN
```

Après avoir lancé ce programme par l'instruction RUN, vous devez choisir et indiquer l'un des modes de fonctionnement. Indiquez par exemple '2' pour obtenir une définition de 640 * 200 points.

Appuyez sur RETURN, des lignes se dessinneront à l'intérieur d'une fenêtre au centre de l'écran. Ces lignes seront séparées par un écart bien défini.
Celui-ci est égal à 20 fois l'épaisseur du trait entre les 10 premières lignes (en partant de la gauche); il sera ensuite décroissant entre les traits suivants et correspondra à 10, 5, 3, 2 et une fois la largeur du trait.

Même si l'on distingue encore bien chaque ligne par intervalle de cinq, nous avons des difficultés à les discerner quand elles sont par trois.
Par deux, il est impossible de les compter sur un moniteur couleur; même si chaque ligne est suivie d'un vide, on voit un bloc uni sur lequel se trouve une curieuse trame.

Appuyons maintenant sur une touche quelconque et passons en mode 1. Là, il est déjà difficile de discerner chaque ligne alors qu'elles ne sont que par groupe de trois. Si l'on passe en mode 0, nous aurons l'impression d'obtenir une surface pleine.

Ces exemples montrent bien ce qu'est la définition.

MOIRE

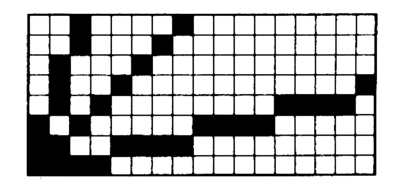
Nous ne sommes pas éloignés de notre premier dessin assisté par ordinateur.

Le scintillement que nous avons obtenu en mode 2 est souvent la base de dessins sur ordinateur, simplement pour les images qui n'ont d'autre but qu'une recherche artistique.

Cet effet se nomme MOIRE - selon un tissu travaillé de telle sorte que sa surface ait un aspect miroitant.
Vous connaissez aussi cet effet de votre téléviseur couleur, quand sur un dessin très fin apparaissent des traits de couleurs qui n'ont rien à y faire.
Sur votre téléviseur tout comme sur notre moniteur, le tube cathodique est responsable de ce phénomène; il a dépassé la limite de sa définition et n'est donc plus en mesure de faire ressortir les détails trop fins.

```
10 REM
20 REM ------------------------------
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:PEN 1:INK 1,0
40 FOR x=1 TO 640 STEP 2
50 PLOT x,1:DRAW x,400
60 NEXT
```

D'autre part, nous savons aussi que l'ordinateur a ses propres limites. Le plus petit point sera toujours un rectangle comme on peut le voir en entrant MODE 0:PLOT 1,1
Une droite ne sera jamais bien droite à l'écran, mais sera représentée en escaliers, comme ci-dessous:



Un faisceau de rayons n'aura pas son apparence, mais donnera un résultat bien plus intéressant:

```
10 REM
20 DEFINT a-z
30 MODE 2
40 FOR x=0 TO 640 STEP 4
50 PLOT 0,0:DRAW x,0
60 PLOT 640,400:DRAW x,0
70 NEXT
80 IF INKEY$="" THEN 80 ELSE END
```

```
10 REM
20 DEFINT a-z
30 MODE 2
40 FOR y=400 TO 0 STEP -3
50 PLOT 0,0:DRAW 640,y
60 PLOT 640,400:DRAW 0,y
70 NEXT
80 IF INKEY$="" THEN 80 ELSE END
```

De petites modifications dans ce programme vous apporteront d'autres graphismes; essayez le mode 1 au lieu du mode 2, ou modifiez le pas après STEP.

Pour créer de tels graphiques sur n'importe quels ordinateurs, il suffit de bien maitriser les coordonnées du système et les instructions permettant l'affichage d'une ligne.

Sur le CPC, nous avons l'instruction PLOT x,y pour placer un point et DRAW x,y pour tracer une ligne du dernier point au point x,y. Les coordonnées x et y répondent a ce que nous avons appris à l'école sur les repères cartésiens; à un détail près il faut le dire, nous ne disposons que du premier cadran et ne pouvons donner que des valeurs positives à x et y.

Sachant tout ceci, nous pouvons créer une multitude de graphiques différents.

Les différentes lignes ne doivent pas avoir leur origine dans les coins de l'écran comme dans l'exemple ci-dessus. Elle peut se situer n'importe où. Les rayons peuvent se croiser; le pas d'une ligne à l'autre peut être variable. Vous pouvez aussi utiliser différentes

couleurs dans un dessin.

Le programme suivant permet le dessin de remarquables graphiques:

```
10 REM GRAPHISME SUR ORDINATEUR
20 REM 1 POINT D'ORIGINE
30 REM ------------------------
40 DEFINT a-z
50 PAPER 0:PEN 1:INK 0,12:INK 1,0:BORDER 12
60 MODE 2
80 X=RND(1)*640:Y=RND(1)*400
90 PAS=RND(1)*10+2
100 FOR X1=1 TO 640 STEP PAS
110 PLOT X,Y:DRAW X1,400
120 PLOT X,Y:DRAW X1,1
130 NEXT X1
140 FOR Y1=400 TO 1 STEP -(PAS/2)
150 PLOT X,Y:DRAW 640,Y1
160 PLOT X,Y:DRAW 1,Y1
170 NEXT Y1
190 FOR I=1 TO 10000:NEXT
200 GOTO 60
```

Pour réaliser des graphiques ayant plusieurs points d'origine, il faut légèrement modifier ce programme

Complétez par une boucle en indiquant en ligne 70 le nombre de points souhaité.
Il faudra peut-être augmenter la valeur du pas pour éviter que votre oeuvre d'art ne se dessine noir sur noir!

```
70 FOR POINT=1 TO 2
180 NEXT POINT
200 CLS:GOTO 70
```

Bien!

Celui qui a lu ce livre, voire qui a suivi les exercices, sait maintenant se servir des couleurs, placer des points ou tracer des lignes.

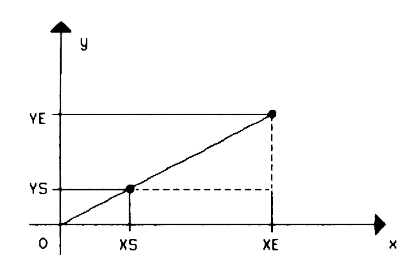
Mais que faire si vous avez besoin d'un cercle? Ou quand la réalisation d'un plan demande de distinguer plusieurs types de lignes identiques et que nous ne disposons en haute résolution graphique que d'une seule couleur?
De même quand vous savez que ce dessin devra sortir sur une imprimante ou table traçante ne disposant également que d'une couleur?

Notre CPC ne nous sera pas d'une grande aide pour résoudre ceci; ayons donc recours aux mathématiques!
Nous ne pourrons pas éviter à partir de maintenant de mettre en parallèle au dessin certaines formules mathématiques.

LIGNES

Elles sont faciles et rapides à tracer avec l'instruction DRAW x,y

Pour les raisons précédentes, il sera souhaitable de développer un algorythme spécifique permettant d'afficher une ligne en pointillé.
Voyons le graphisme suivant:



Nous voulons relier deux points se trouvant dans notre graphique.Nous

ne connaissons que leurs coordonnées x et y.

La distance d'un point par rapport à l'autre ne peut être donnée séparément que pour chaque axe:

dx = xe - xs

dy = ye - ys

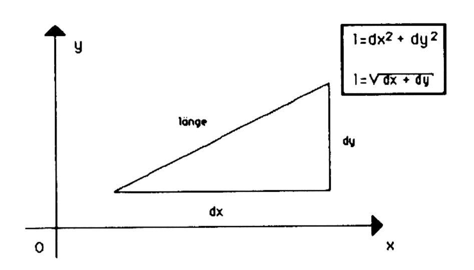
Nous ne pouvons pas lire quel est le plus court chemin pour tracer la ligne voulue entre ces deux points.

Souvenons-nous d'un ancien savant, de Pythagore, et prenons le théorème connu selon lequel dans un triangle rectangle, le carré de l'hypoténuse est équivalent à la somme des carrés des deux autres côtés.

Pour notre exemple, la distance cherchée correspond justement au calcul de l'hypoténuse, soit:

longueur = SQR(dx*dx+dy*dy)

Nous connaissons maintenant le nombre de points nécessaires pour tracer cette ligne; c'est là un pas important.



Pour définir les coordonnées de chaque point, divisons cette longueur par la différence tout d'abord calculée. Nous saurons ensuite

exactement quelle valeur ajouter aux coordonnées de notre point de départ:

part x = dx / longueur

part y = dy / longueur

== x = xs + PAS * dx

y = yb + PAS * dy

PAS correspond à la droite à tracer, ici seulement un point, et dépend donc du pas de la boucle (qui est elle-même fonction de la densité des points de la ligne que vous souhaitez obtenir).

Ces considérations peuvent être mises en forme dans le programme suivant:

```
10 REM ROUTINE D'AFFICHAGE D'UNE LIGNE
20 MODE 2:PEN 1:INK 1,24:BORDER 0:INK 0,0
30 DEFINT A-Z
40 INPUT "ORIGINE X ";XS
50 INPUT "ORIGINE Y ";YS
60 INPUT "VERS POINT X ";XE
70 INPUT "VERS POINT Y ";YE
80 INPUT "TRACER UN POINT SUR...";SW
90 DX=XE-XS:DY=YE-YS
100 LONG=SQR(DX*DX+DY*DY)
110 FOR DISTANCE =0 TO LONG STEP SW
120 PLOT XS+DISTANCE*DX/LONG,
    YS+DISTANCE*DY/LONG
130 NEXT DISTANCE
```

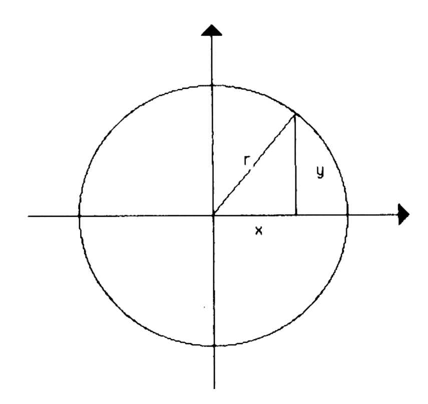
Vous pouvez ainsi tracer toute ligne pointillée.

CERCLES

Pour la création de dessins, en dehors de lignes, nous utiliserons très souvent des arcs de cercles.

Le BASIC de base de notre CPC ne disposant pas de l'instruction très pratique CIRCLE, ici encore, nous allons être obligé de passer par une formule mathématique.

Sur un plan, un cercle est constitué de points tous disposés à une distance égale d'un même point, le centre du cercle.
La distance du centre à l'un des points périphériques est le rayon.



En algèbre, et toujours selon Pythagore, le rayon peut s'exprimer comme suit:

r * r = x * x + y * y

r = SQR(x * x + y * y)

Ceci en prenant comme coordonnées de base 0,0, soit le point d'intersection des axes x et y.
Cette équation peut aussi être posée pour y:

r * r = x * x + y * y

y = SQR(x * x + y * y)

Pour afficher un cercle à l'écran, nous devrons donner pour x successivement toutes les valeurs de -r à r. Une boucle sera la mieux adaptée à cette opération:

```
10 MODE 2
20 DEFINT A-Z
30 ORIGIN 320,200
40 INPUT "RAYON ";RAYON
50 FOR X=-RAYON TO RAYON
60 Y=SQR(RAYON*RAYON-X*X)
70 PLOT X,Y:PLOT X,-Y
80 NEXT X
```

En pratique, cette routine pose quelques problèmes; pour des rayons importants, le pas est trop grand et ne permet pas d'obtenir une ligne fermée.
Pour ceci, ajouter à la ligne 50 l'instruction STEP .1; le programme sera beaucoup plus lent puisque les calculs seront multipliés par dix.

Une autre solution sera plus appropriée. Il ne faut pas calculer le positionnement de chaque point en fonction de l'axe X, mais avancer par 'degré' sur le périmètre et ainsi calculer directement les coordonnées x et y.
Pour ceci, nous devons quelque peu devancer un prochain chapitre et oublier notre système de coordonnées pour prendre en considération le système de coordonnées polaires.
Ce raisonnement permet de définir l'emplacement d'un point sur un plan à l'aide d'une part de la distance de celui-ci par rapport à un point d'origine, et d'autre part l'angle entre la ligne représentant cette distance et l'axe positif x.

Cette façon de faire est idéale pour nous. Les données pour le point 'p' seraient réduites à 0 degré et longueur = 50. Pour obtenir les autres points, il suffirait seulement d'augmenter la valeur angulaire d'un point jusqu'à 360 degrés.

Et pour obtenir des coordonnées x/y, nous utiliserons les deux équations suivantes:

x = r * cos(theta)

y = r * sin(theta)

Intégré au programme suivant, nous pourrons tracer n'importe quel cercle:

```
10 MODE 2
20 DEFINT A-Z
30 ORIGIN 320,200
40 DEG
50 INPUT "RAYON ";RAYON
60 FOR DEGRE=1 TO 360
70 X=RAYON*COS(DEGRE)
80 Y=RAYON*SIN(DEGRE)
90 PLOT X,Y
100 NEXT DEGRE
```

ELIPSES

Elles sont similaires aux cercles. Les routines précédentes pourront aussi être utilisées.

Pour dilater ou compresser un cercle, il suffit d'adjoindre un facteur complémentaire à nos formules, soit:

```
10 MODE 2
20 DEFINT A-Z
```

```
30 ORIGIN 320,200
40 DEG
50 INPUT"rayon";rayon
54 INPUT"variation axe des x (entre 0 et 10)";vx
58 INPUT"variation axe des y (entre 0 et 10)";vy
60 FOR degre=1 TO 360
70 x=rayon*COS(degre)*vx
80 y=rayon*SIN(degre)*vy
90 PLOT x,y
100 NEXT degre
```

VX et VY sont des facteurs qui entrainent la déformation du cercle dans l'axe correspondant.

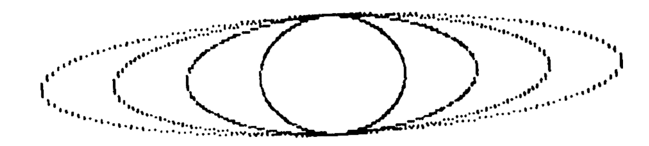
Ne donnez pas de valeur trop élevée à ces variables; une valeur de 0 à 10 serait appropriée.

```
10 REM elipses
20 REM -------
30 MODE 2
40 DEFINT a-z
50 ORIGIN 320,200
60 DEG
70 IF w THEN CLS
80 CLS
90 INPUT "rayon ";rayon
100 INPUT"déformation en direction
    des x ou y (x/y)";direction$
110 IF LOWER$(direction$)="x"THEN vx=-1
120 IF LOWER$(direction$)="y"THEN vy=-1
130 FOR degré=1 TO 360
140 FOR facteur=0.5 TO 3.5 STEP 0.5:REM
    degré de déformation
150 x=rayon*COS(degré)
160 IF vx THEN x=x*facteur
170 y=rayon*SIN(degré)
```

```
180 IF vy THEN y=y*facteur
190 PLOT x,y
200 NEXT facteur
210 NEXT degré
220 FOR i=1 TO 10000:NEXT
230 w=-1:GOTO 70
```

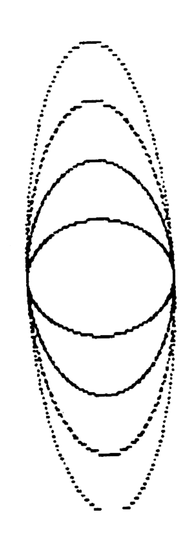
RAYON: 50

x VX



RAYON: 50

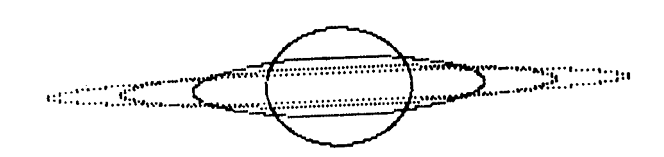
x VY



RAYON: 50

x VX

/ VY



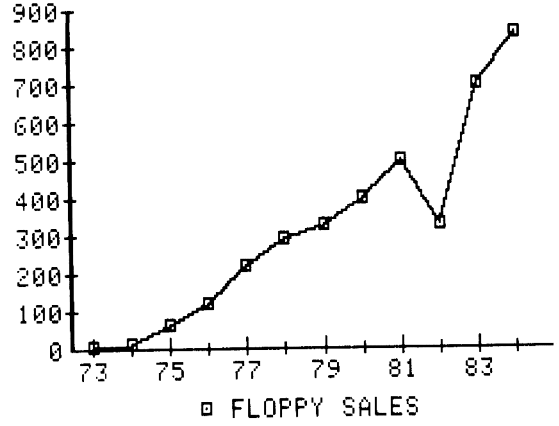
Utilisation: histogrammes

Un schéma vaut plus qu'une longue explication, c'est également vrai dans les affaires. Les dirigeants de sociétés ont toujours sous la main les graphiques montrant leur part de marché. Selon le contexte et le but recherché, les statistiques seront mises sous forme d'un graphisme approprié pour comprendre celui-ci sans le regarder trop longuement ou trop réfléchir.

Pour mettre en évidence certaines tendances, les histogrammes en courbe sont les plus adaptés.

L'exemple type pourrait être la comparaison des chiffres d'affaires mensuels ou annuels d'une société.

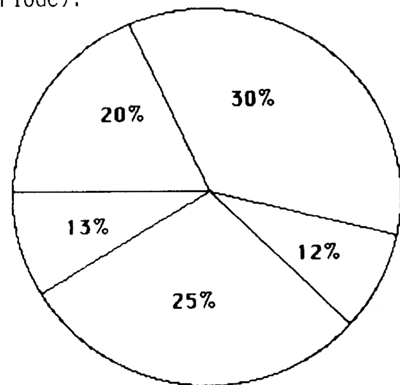
L'évolution de la courbe permet de tirer des conclusions sur l'exercice considéré ainsi que des prévisions sur la prochaine période.



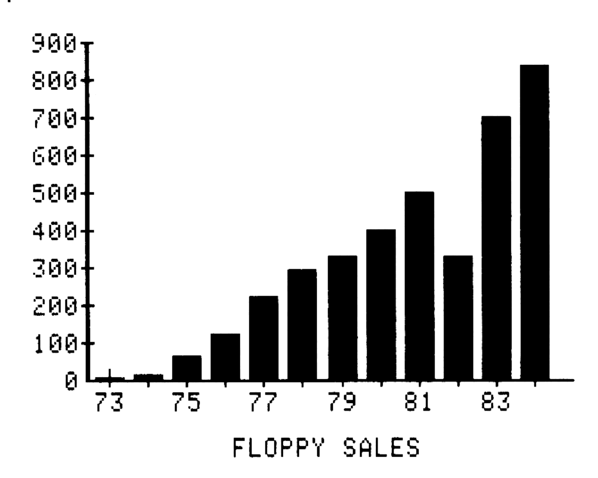
Les histogrammes en 'camembert' et par colonne seront moins efficaces pour ceci et conviendront mieux à la comparaison avec une ou plusieurs données différentes.

Leur nombre est naturellement limité; un tableau de 10 mètres et 500 colonnes serait certes très impressionnant, mais peu expressif. Ce type de représentation est donc utile pour rendre des données correspondant elles-mêmes à une plage de temps assez large.

Ici aussi, les données à représenter (chiffres d'affaires) sont fonction d'un autre facteur (période).



Il en est tout autrement pour les représentations en 'camembert'. L'important n'est ici pas la valeur statistique puisque nous ne pouvons partager le cercle selon un pourcentage. L'intérêt est bien plus dans la visualisation de la répartition de valeurs indépendantes les unes des autres.
Ce n'est pas pour rien que l'on dit que chacun veut avoir sa part du gateau!
Voyons la représentation des parts de marché de différentes sociétés d'un secteur.



Mais quels sont les pas à effectuer pour créer à l'écran un tel graphique?

Il faut faire un cercle et définir dans les bonnes proportions les différents secteurs. Enfin, il faudrait inscrire la légende sur chaque 'part'; sans ceci le diagramme serait difficile à exploiter.

Les données devant être transmises au programme, les grandes lignes seraient:

initialisation

entrée des données

traçage des points

inscription des légendes

L'initialisation devrait être claire pour toute personne ayant lu le manuel d'utilisation du CPC; vous savez tous que la place mémoire des variables doit être réservée, tout comme il est connu que des données similaires sont plus faciles à entrer par l'intermédiaire d'une boucle. Nous n'entrerons donc pas dans le détail du programme suivant:

```
30 MODE 1:PAPER 0:INK 0,12:BORDER 12:PEN 1
   :INK 1,0
40 DEG
50 DIM valeur(20),degré(20)


1000 MODE 1:PRINT" entrée des données"
1010 PRINT:INPUT"combien de données ";nb
1020 PRINT:total=0
1030 FOR I=1 TO nb
1040 PRINT I;:INPUT valeur(I)
1050 total=total+valeur(I)
```

```
1060 NEXT I
```

Ceux qui ont lu les pages précédentes savent tous comment dessiner un cercle et tous ceux qui savent faire une règle de trois ou calculer des proportions feront comme nous pour faire le partage en secteurs.

Les 360 degrés d'un cercle correspondent à 100%, c'est à dire à la somme totale des valeurs que nous allons indiquer.
Une donnée correspondra donc à autant de degrés que la proportion qu'elle aura de la somme globale:

```
 somme partielle       angle du secteur
------------------ = -------------------
  somme totale            360 degrés
```

soit, pour le secteur recherché:

```
4000 MODE 2:ORIGIN 320,200
4030 FOR I=1 TO nb
4040 degré(I)=INT(valeur(I)*360/total)
4050 NEXT I
```

Le tableau 'degré (I)' comprend maintenant la valeur des différents secteurs.

Il ne nous reste plus qu'à vérifier à chaque affichage d'un point (PLOT) si la fin d'un secteur est atteinte (4100).
Si c'est le cas, une ligne sera tirée de ce point au centre du cercle, et la valeur angulaire calculée à ce moment sera additionnée au prochain secteur afin d'être certain qu'il ne soit tenu compte que de ce nouveau secteur:

```
4060 r=150:I=1
4070 FOR degré=1 TO 360
4080 x=r*COS(degré):y=r*SIN(degré):PLOTx,y
```

```
4100 IF degré(I)=degré THEN DRAW 0,0:
     degré(I+1)=degré(I+1)+degré(I):i=i+1
4110 NEXT degré
4130 IF INKEY$=""THEN4130
4140 GOTO 100
```

L'inscription de la légende pourrait être faite à la fin d'un secteur (4100), mais cela ne serait pas très clair - à quel secteur cela correspondrait-il? -
D'autre part, ce texte ne serait pas toujours bien positionné, une fois dans le diagramme et une fois en dehors!

Il serait souhaitable d'inscrire le texte au milieu du secteur; nous en tiendrons compte lors du calcul des valeurs angulaires de chaque secteur:

```
4040 degré(I)=INT(valeur(I)*360/total):
     postexte(I)=degré(I)/2
4060 r=150:I=1:vieutexte=0
4090 IF vieutexte+postexte(I)=degré THEN
     GOSUB 5000
```

Quand il le faudra, le sous-programme en ligne 5000 sera appelé; il calcule tout d'abord la longueur du titre.
Il vérifie ensuite si l'inscription doit se faire sur la moitié gauche du cercle (x 0) ou sur celle de droite.
Dans le premier cas, le curseur sera placé sur la gauche d'un blanc plus la longueur de l'inscription alors que dans le deuxième cas, le curseur sera décalé d'un blanc de la ligne du cercle:

```
5000 TAG:x1=(LEN(titre$(I))+1)*8
5010 IF x 0 THEN MOVER 10,0
5020 IF x 0 THEN MOVER -x1,0
5030 IF y 0 THEN MOVER 0,10
```

Lors de la sortie de texte en mode graphique, le point d'origine sera toujours le point supérieur gauche d'un caractère, ce qui donne un décalage supplémentaire dans le cas où y 0.

Tenez également compte du point virgule de la ligne 5050. Il est indispensable pour éviter, dans ce mode d'affichage, la sortie du sigle graphique correspondant au RETURN.

Bien entendu, avant d'écrire le texte, vous devez l'indiquer au programme.

Il serait peut-être intéressant de mémoriser certaines données.

Le listing suivant reprend un programme aussi complet:

```
10 REM camembert
20 DEFINT a-z
30 MODE 1:PAPER 0:INK 0,12:BORDER 12:PEN 1:INK 1,0
40 DEG
50 DIM valeur(20),degre(20)
100 MODE 1:PRINT"      ****camembert****     ":
    PRINT:PRINT" 1 - entrer nouvelles donnes"
110 PRINT" 2 - sauvegarder donnees"
120 PRINT" 3 - charger donnees"
130 PRINT" 4 - dessin"
140 PRINT:PRINT" faites votre choix
150 a$=INKEY$:IF a$="" THEN 150 ELSE a=VAL(a$)
160 IF (a>4 OR a<1) THEN 100
170 ON a GOTO 1000,2000,3000,4000
990 REM *********** entree des donnees
1000 MODE 1:PRINT "entree des donnees"
1010 PRINT:INPUT "combien de donnees";nb
1020 PRINT: total=0
1030 FOR i=1 TO nb
1040 PRINT i;:INPUT valeur(i)
1050 total=total+valeur(i)
1060 NEXT i
1070 PRINT:INPUT "le graphisme doit-il avoir
     un titre ?";e$
```

```
1080 IF LEFT$(LOWER$(e$),1)="o" THEN INPUT "titre ";
     titre$
1090 GOTO 100
1990 REM ******************sauvegarde des donnees
2000 CLS:PRINT"d'accord avec le nom du fichier ";
     LEFT$(titre$,8):INPUT e$
2010 IF LEFT$(LOWER$(e$),1)="n" THEN INPUT"nom du
     fichier ";dn$
2020 OPENOUT dn$
2030 PRINT#9,titre$
2040 PRINT#9,nb
2050 FOR i=1 TO nb
2060 PRINT#9,valeur(i):PRINT#9,titre$(i)
2070 NEXT
2080 CLOSEOUT
2090 GOTO 100
2990 REM *********** chargement des donnees
3000 CLS:INPUT"nom du fichier ";dn$
3010 OPENIN dn$:total=0
3020 INPUT#9,titre$
3030 INPUT#9,nb
3040 FOR i=1 TO nb
3050 INPUT#9,valeur(i):total=total+valeur(i):
     INPUT#9,titre$(i)
3060 NEXT i
3070 CLOSEIN
3080 GOTO 100
3990 REM ************** dessin
4000 MODE 2:ORIGIN 320,200
4010 IF nb=0 THEN PRINT"***erreur ! pas de donnees":
     FOR i=1 TO 10000:NEXT GOTO 100
4020 LOCATE 1,1:PRINT"diagramme: ";titre$
4030 FOR i=1 TO nb
4040 degre(i)=INT(valeur(i)*360/total)
4050 NEXT i
4060 r=150:i=1:vieutext=0
4070 FOR degre=1 TO 360
4080 x=r*COS(degre):y=r*SIN(degre):PLOT x,y
4085 IF vieutext+postext(i)=degre THEN GOSUB 5000
```

```
4090 IF degre(i)=degre THEN vieutext=degre:PLOT x,y:
     DRAW 0,0:degre(i+1)=degre(i+1)+degre(i):i=i+1
4100 NEXT degre
4120  GOSUB 5000
4130 IF INKEY$="" THEN 4130
4140 GOTO 100
4990 REM ********** label
5000 TAG:x1=(LEN(titre$(i))+1)*8
5010 IF x>0 THEN MOVER 10,0
5020 IF x<0 THEN MOVER -x1,0
5030 IF y>0 THEN MOVER 0,10
5040 PRINT titre$(i);
5050 TAGOFF
5060 RETURN
```

formes pleines

Elles peuvent être réalisées à l'aide des routines précitées. Précédemment, nous avons calculé les coordonnées servant au tracé du cercle que nous souhaitions obtenir. Plutôt que d'afficher uniquement ce point, il suffit de le relier au centre par l'instruction DRAW. Nous obtiendrons ainsi une surface circulaire pleine souvent appelée en informatique 'disc'.

Utilisons la dernière routine que nous avons développée; le manque de précision est grand. La routine employant le théorème de Pythagore sera idéale.
L'arc de cercle sera ici calculé en fonction de l'axe des x, nous obtiendrons avec DRAW une surface bien pleine.

```
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:
   PEN 1:INK 1,0
40 DEFINT A-Z
49 REM définir l'origine des coordonnées
50 ORIGIN 320,200
60 DEG


999 REM entrée de la définition du cercle
1000 LOCATE 1,1:INPUT" rayon ";rayon
1310 FOR X=-rayon TO rayon
1320 Y=SQR(rayon*rayon-X*X)
1330 PLOT X,Y:DRAW x,-Y
1340 NEXT X
1360 GOTO 1000
```

La surface pleine d'un cercle de diamêtre quelconque apparait relativement vite sur l'écran.

Bien entendu, l'algorythme ici employé implique que l'origine du système de coordonnées soit identique au centre du cercle.

En informatique, il est en fait courant de procéder par addition pour faire évoluer chaque fonction dans un système de coordonnées. Le programmeur du CPC 464 devrait se simplifier les choses en imposant le centre du cercle comme origine du système de coordonnées.

Il ne devra pas oublier de remettre l'origine standard (souvent 0,0 ou 320,200) après la routine:

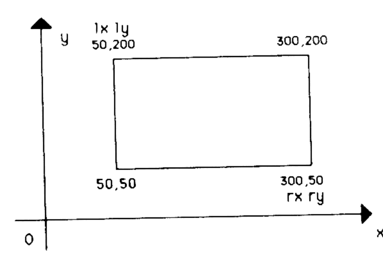
```
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:
   PEN 1:INK 1,0
40 DEFINT A-Z
49 REM définir l'origine des coordonnées
50 ORIGIN 0,0
60 DEG


999 REM entrée de la définition du cercle
1000 LOCATE 1,1:INPUT" rayon ";rayon
1010 INPUT" centre (x,y) ";mx,my
1299 REM--------------DISC
1300 ORIGIN mx,my:REM nouveau point d'origine
1310 FOR X=-rayon TO rayon STEP 1
1320 Y=SQR(rayon*rayon-X*X)
1330 PLOT X,Y:DRAW X,-Y
1340 NEXT X
1350 ORIGIN 0,0:REM rétablir point d'origine
1360 GOTO 1000
```

Si vous souhaitez obtenir des rayures, aucun problème, cela peut s'obtenir en ligne 1310 en augmentant la valeur du pas STEP.
Seul le pas de '1' permettra d'obtenir une surface pleine.

Rectangles

Ce qui vaut pour un cercle devrait être assez bien pour tracer un rectangle. Mais cela n'est pas le cas, nous devons donc trouver une méthode simple pour faire ce dessin.
Bien entendu, quatre droites perpendiculaires nous mèneraient aussi au but, mais pourquoi s'embêter avec toutes ces coordonnées alors que l'on peut s'en passer?



Vous voyez qu'il est simple de trouver et d'indiquer les coordonnées manquantes (lx,ry et rx,ly).
Puisque cela est si facile, de nombreux constructeurs d'ordinateurs ont prévu au répertoire du BASIC l'instruction BOX.
Après lui avoir indiqué les coordonnées des points haut à gauche et bas à droite, celle-ci affiche en un clin d'oeil un rectangle.

Notre AMSTRAD ne connait pas cette instruction. Le petit programme suivant nous donne quand même cette facilité:

```
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:
   PEN 1:INK 1,0
40 DEFINT A-Z
49 REM origine des coordonnées
50 ORIGIN 0,0
1399 REM dessin rectangle
1400 LOCATE 1,1:INPUT " COORDONNEES HAUT/
```

```
    GAUCHE (X,Y)";LX,LY
1410 INPUT" BAS/ DROITE (X,Y) ";RX,RY
1440 PLOT LX,LY:DRAW RX,LY:DRAW RX,RY:DRAW
    LX,RY:DRAW LX,LY
1490 GOTO 1400
```

La partie vidéo du CPC ayant été conçue spécialement pour le graphisme, la rapidité d'éxécution est pratiquement identique aux ordinateurs faisant appel à une routine d'interprétation.
Vous devez seulement faire attention que les coordonnées des points soient bien en diagonale.

Pour obtenir à l'écran un rectangle plein, en Anglais FILLED BOX, nous pouvons aussi utiliser un deuxième procédé.

Il faut faire comme pour le dessin d'un cercle, se déplacer le long d'un axe et tracer des lignes de la longueur voulue dans l'autre axe.

```
30 MODE 2:PAPER 0:INK 0,12:BORDER 12:
   PEN 1:INK 1,0
40 DEFINT A-Z
49 REM origine des coordonnées
50 ORIGIN 0,0
1399 REM dessin rectangle
1400 LOCATE 1,1:INPUT " COORDONNEES HAUT/
    GAUCHE (X,Y)";LX,LY
1410 INPUT" BAS/ DROITE (X,Y) ";RX,RY
1420 INPUT" REMPLIR (O/N)";E$
1430 IF UPPER$(E$)="O" THEN 1500
1440 PLOT LX,LY:DRAW RX,LY:DRAW RX,RY:DRAW
    LX,RY:DRAW LX,LY
1490 GOTO 1400
1499 REM FILLED BOX
```

```
1500 REM
1510 FOR Z=LX TO RX STEP 1
1520 PLOT Z,LY:DRAW Z,RY
1530 NEXT Z
1550 GOTO 1400
```

Tant que nous nous en tenons à la suite 'gauche - haut / bas - droite', tout fonctionne bien. Dans le cas contraire, rien ne se fait.

Cela vient de la boucle puisque celle-ci devrait diminuer la valeur 'x' à chaque passage au lieu de l'augmenter.

Une solution pourrait être donnée par le contrôle de la deuxième valeur; si celle-ci est inférieure à la première, les valeurs pourront être interchangées.

Complétez donc le programme ci-dessus par la ligne suivante:

```
1500 IF LX>RX THEN HX=RX:RX=LX:LX=HX
```

La deuxième solution pour réaliser des surfaces pleines exploite la technique des fenêtres (WINDOWS) qui est très facile à utiliser sur le CPC.

Fenêtres

Elles datent en fait d'une époque à laquelle certains constructeurs d'ordinateurs et éditeurs de programmes astucieux ont voulu simplifier la tâche aux secrétaires tout à fait dépassées par la technique informatique. Il fallait faire entrer le nouveau collègue 'ordinateur' dans les moeurs!
Car bien souvent, même après formation, les utilisateurs passaient plus de temps à compulser le manuel d'utilisation qu'à travailler.

L'idée d'afficher à l'écran les instructions comme sur une feuille du mode d'emploi à donc vu le jour.

Tour comme l'on peut mettre une feuille de coté, il devait être possible de retirer cet affichage de l'écran.

Pour arriver à un résultat aussi complet, le niveau de programmation est déjà assez important, la place mémoire du système est aussi mise à forte contribution: avant d'afficher une fenêtre, il faut en effet mettre en mémoire le contenu de l'écran.
Si la mémoire écran est définie comme sur le CPC, un affichage de quatre fenêtres superposées pouvant être chacune rapelée demanderait 48 KO de mémoire.
C'est aussi la raison pour laquelle nous ne trouvons ces caractéristiques que sur des systèmes disposant d'une mémoire de masse importante, soit un disque dur, soit une mémoire vive de plusieurs centaines de kilo-octets.

Notre CPC ne répond ni à l'un, ni à l'autre de ces critères. Malgré cela, la fonction WINDOW du BASIC intégré permet la programmation de certaines applications intéressantes.

Bien qu'il s'agisse toujours de rectangles ou carrés, la définition de nos fenêtres se différencie légèrement des procédures vues plus haut.
Après l'instruction 'WINDOW' elle-même suivie de ' ', il faut indiquer un chiffre de 1 à 7 correspondant à un canal attribué à cette fenêtre qui pourra ainsi toujours être addressée par son intermédiaire.
Ensuite sont introduits successivement les emplacements écran des coins gauche, droite, haut et bas.
Ces chiffres peuvent donc aller de 1 à 22 (en mode 0), mais aussi jusqu'à 80 (en mode 2).

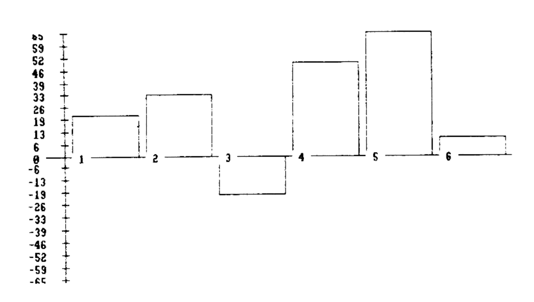
L'instruction 'PRINT canal' affichera les données dans la fenêtre correspondante au numéro de canal, tout comme 'CLS canal' effacera

qui y est affectée.

```
10 MODE 0
20 DROITE=22:BAS=24
30 COULEUR=2
40 FOR CANAL=0TO7
50 GAUCHE=GAUCHE+2:HAUT=HAUT+2
60 WINDOW#CANAL,GAUCHE,DROITE,HAUT,BAS
70 COULEUR=COULEUR+1
80 PAPER #CANAL,COULEUR
90CLS #CANAL
100 FOR I=1TO400:NEXT
110 NEXT
```

Utilisation: histogrammes en colonne

Il est maintenant temps de complèter notre programme 'pieplot' par une représentation graphique en colonne:



En principe, nous sommes confrontés à des problèmes identiques à ceux que nous avons rencontrés précédemment; les données doivent être saisies puis adaptées à la surface dont nous disposons pour la représentation graphique, c'est à dire mises à une certaine échelle.

Cette mise à l'échelle est très importante quelque soit le type de représentation. Nous allons donc examiner plus précisément cette procédure.

La première opération est en fait toujours une comparaison, soit toutes les valeurs sont dirigées vers une barrière maximum comme c'est le cas pour un 'camembert' (maxi = 360 degré), soit la donnée la plus importante sera prise en référence et donc en valeur maximum par rapport a toutes les autres données.

Pour avoir un diagramme occupant la plus grande surface possible, cette valeur maximale sera mise à égalité avec la surface disponible:

PLOTMAX

DATAMAX

Le quotien de cette opération sera en réalité le facteur de mise à l'échelle permettant par simple multiplication d'étalonner les autres données par rapport à la valeur maxi et donc à la place disponible pour le diagramme.

Le produit 'donnée * échelle' donne directement la distance de la valeur par rapport à la ligne zéro.

Les coordonnées de la ligne zéro étant choisie dès le début, la distance du tracé étant calculée et deux points seulement étant nécessaires pour faire ce tracé, notre problème est résolu. Le programme 'camembert' peut être complèté:

```
50 DIM valeur(20),degré(20),bvaleur(20)

100 MODE 1:PRINT"    * * * *  DATAPLOT  *
    * * *":PRINT:PRINT" 1 -entrer nouvelles
    données"
135 PRINT" 5 - diagramme à colonnes"
160 IF (a>5 OR a<1) THEN 100
170 ON a GOTO 1000,2000,3000,4000,6000
```

Ces lignes réservent la place pour les variables et ajoute au menu l'option pour la représentation de diagrammes à colonnes.
La ligne 1050 est indispensable pour noter la valeur maxi et calculer comme nous l'avons expliqué le facteur d'échelle.

```
1050 total=total+valeur(i):vmax=MAX
     (i-1),valeur(i))

6000 MODE 2:mfacteur=199/vmax:rx=60:écart=10
```

Vous vous étonnez probablement de notre calcul du rapport d'échelle basé sur 199 points et non 400 (dans l'axe y).
Nous devons tenir compte de la représentation éventuelle de valeurs négatives, d'où la division par deux du nombre maximum de points. D'autre part, il serait possible d'obtenir une valeur légèrement supérieure à 200 lors de la multiplication avec 'mfacteur', le trait supérieur ne serait donc pas tracé.

Pour permettre une inscription ultérieure sur l'axe 'y', il serait même indiqué de réduire 'mfacteur'.

Pour tenir compte de tout ceci, l'axe 'x' ne pourra être qu'au centre de l'écran. L'axe 'y' sera légèrement décalé du bord gauche de l'écran pour laisser place à un marquage.

```
6060 ORIGIN 0,200
6070 PLOT 10,0:DRAW 640,0:REM axe 'x'
6080 PLOT 50,-200:DRAW 50,200:REM axe 'y'
```

Avant de dessiner les différentes colonnes, nous devons en calculer la taille à l'intérieur d'une boucle. Leur largeur sera également fonction de leur nombre, tout comme de l'écart entre chaque colonne et naturellement de la place écran disponible.

```
6030 FOR i=1 TO nb
6040 bvaleur(i)=valeur(i)*mfacteur
6050 NEXT i

6090 lcol=580/nb-écart
```

Les colonnes seront représentées à l'aide d'une des routines précedemment mise au point:

```
6100 FOR i=1 TO nb
6110 ry=0:lx=rx+lcol:ly=bvaleur(i)
6120 PLOT lx,ly:DRAW rx,ly:DRAW rx,ry:
     DRAW lx,ry:DRAW lx,ly
6140 rx=rx+écart+lcol:REM début prochaine
     colonne
6150 NEXT i
```

Ensuite, il faudrait en tout cas intituler l'axe 'y'. Ceci demande encore un peu de réflexion et quelques calculs.

Il ne s'agit en effet pas uniquement d'inscrire les intitulés avec un écart constant, mais aussi de faire cet affichage au niveau de chaque colonne. Il en résulte donc déjà une multiplication avec 'mfacteur'. La valeur maxi sera identique à 'mvaleur'; une portion aura la taille de 'mvaleur' divisé par le nombre souhaité de portions, ici '10':

```
6010 TAG
6220 FOR y1=0 TO wmax STEP wmax/10
6230 PLOT 55,y1*mfacteur:DRAW 45,y1*mfacteur
6240 MOVE 1,y1*mfacteur+4
6250 PRINT INT(y1);
6260 NEXT y1
```

Il est bien sur possible de faire cette opération pour la partie négative de l'axe 'y':

```
6170 FOR y1=-wmax TO 0 STEP wmax/10
6180 PLOT 55,y1*mfacteur:DRAW 45,y1*mfacteur
6190 MOVE 1,y1*mfacteur+4
6200 PRINT INT(y1)
```

```
6210 NEXT y1
6270 TAGOFF
6280 GOTO 4130
```

Si vous voulez également mettre une légende sur l'axe des 'x', complètez le programme par la ligne suivante:

```
6130 MOVE rx,ry+4:PRINT i;
```

Vous avez maintenant un programme vous proposant la représentation d'une série de données soit sous la forme d'un histogramme en colonnes, soit en 'camembert'.
La mise à l'échelle se fera toujours pour utiliser la plus grande surface utile de l'écran.

Mais n'oubliez pas que le meilleur programme ne peut contourner les caractéristiques physiques d'un écran, alors n'essayez pas d'introduire un trop grand nombre de données.

Chapitre 3

TECHNIQUES DU GRAPHISME

caractéristiques techniques de l'écran

Celles-ci sont déterminantes pour décider d'une éventuelle utilisation graphique de votre ordinateur.
La.représentation graphique de fonctions mathématiques demandera par exemple une haute définition graphique; la couleur ne jouant pas d'importance, l'écran peut être monochrome. Par contre dans le cadre de jeu d'arcade, la couleur est indispensable, le graphisme peut être légèrement moins précis mais doit permettre un affichage très rapide de l'image graphique à l'écran.

Pour cette raison, on s'efforce d'exploiter au mieux les possibilités de notre ordinateur, afin de n'utiliser un minimum de formules ralentissant elles-mêmes l'exécution des opérations.
Sur le CPC 464, l'instruction 'PRINT H' s'exécutera bien plus rapidement que si l'on effectue le tracé de trois lignes redonnant la même image que cette lettre H.
La différence du temps d'affichage s'explique simplement : dans le premier cas le système d'exploitation recherche à un endroit précis de la mémoire la définition du caractère, alors que dans l'autre méthode, deux instructions 'PLOT' et trois 'DRAW' seront à exécuter.

Au cours des dernières années, trois techniques différentes se sont imposées et utilisent toutes les trois des caractères prédéfinis.

sprites

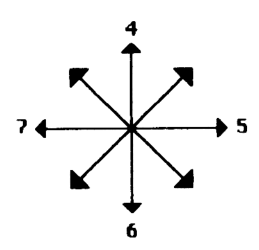
Il s'agit là du plus récent développement qui a certainement beaucoup aidé au succès incontesté de l'ordinateur individuel Commodore 64.

Cet ordinateur, qui permet de reproduire sur son poste de télévision des jeux dans la même qualité que dans les salles de jeux; un circuit intégré de gestion du graphisme permet à l'utilisateur de se constituer et de déplacer une série de figures pouvant avoir jusqu'à 24 fois 21 points.

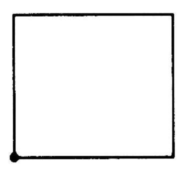
L'un des principaux avantages des 'SPRITES' est en fait leurs possibilités de se déplacer sur l'écran sans déteriorer celui-ci, s'est-à-dire sans effacer ou écrire quoique ce soit sur celui-ci.

SHAPES

Ce sont les ancètres des 'SPRITES'. Leur forme est bien définie par le constructeur de l'ordinateur et pourra donc être différent d'un ordinateur à l'autre. En principe il s'agira néanmoins toujours de la silhouette d'une figure à représenter.
Celles-ci seront ensuite notées dans un tableau, tableau des 'SHAPES', en fonction des différentes directions indiquées pour le curseur.



Il y aura donc un code spécifique pour un déplacement sur la gauche, sur la droite, vers le haut ou vers le bas ; des codes correspondant au déplacements diagonaux sont souvent disponibles tout comme d'autres permettant de définir si un trcé doit être effectué ou non pendant le déplacement.



Si, d'autre part, il est convenu d'un point d'origine; par exemple que le dessin d'une figure commence toujours dans le coin bas et gauche, le carré ci-dessus pourra être défini comme suit:

4,5,6,7

Cette méthode n'est pas forcément la plus rapide, mais permet un certain nombre d'opérations telles que rotation ou agrandissemant.
Pour cette raison nous expliquerons une méthode dans le prochain chapitre permettant d'utiliser cette technique sur l'Amstrad CPC.

caractères

Si les 'SHAPES' sont les ancêtres des 'SPRITES', les 'CARACTERES' et leurs combinaisons, les 'CHAINES', sont l'équivalent d'Adam et Eve pour les hommes, c'est-à dire l'origine de toute existence et technique.

Au début de l'utilisation des ordinateurs, on était très content de disposer d'une imprimante et pas uniquement d'un appareil sortant des cartes ou bandes perforées devant tout d'abord être décodées.
Mais comme ces imprimantes n'étaient en fait que des telex adaptés aux besoins informatiques, leurs polices de caractères étaient très limitées.

Bien entendu, il était déjà possible de mettre sur papier des graphiques.
Malheureusement le plus petit point de ces graphiques avait la dimension d'un caractère classique, qu'il s'agisse d'une étoile ou d'un signe de ponctuation.

Ce n'est qu'en 1953 qu'a été créé le premier affichage de point à l'écran, qui soit aussi la base des procédés d'affichage de caractères actuellement utilisés.
Les images à représenter, et donc aussi les caractères, étaient divisés en un certain nombre de petits points allumés successivement par le rayon cathodique.

Le graphisme par matrice n'a pu être remplacé jusqu'à ce jour. Les seules améliorations qui on pu y être apportées furent l'emploi d'un plus grand nombre de points permettant la représentation d'un caractère.

La majorité des ordinateurs de jeux et familiaux ont actuellement une matrice de définition de caractères de 8*8 points, soit 64 points.

Il s'agit là d'un bon conpromis; chaque matrice de caractères devant être mise en mémoire à un endroit bien défini de celle-ci, une définition plus grande occuperait plus de mémoire, entrainant donc des solutions techniques plus onéreuses.
D'autre part, le tube cathodique doit également être en mesure d'afficher clairement chaque point. Le côté précision de cet affichage ne devant pas être pris à la lettre:
Si tel était le cas, chaque point resterait visible et ne donnerait pas l'impression d'une ligne ou d'un trait continu.

Durant de longues années, on était obligé d'utiliser les caractères intégrés non seulement pour les traitements de textes, mais aussi pour la représentation des différents graphiques.
Quelques artistes informaticiens précoces se sont même fait le travail de transposer des photographies en lettres et d'en faire manuellement la digitalisation.
Ils utilisent la densité de point d'un caractère pour parfaire cette image; un 'm' est en effet bien plus dense qu'un 'c' ou même un point.

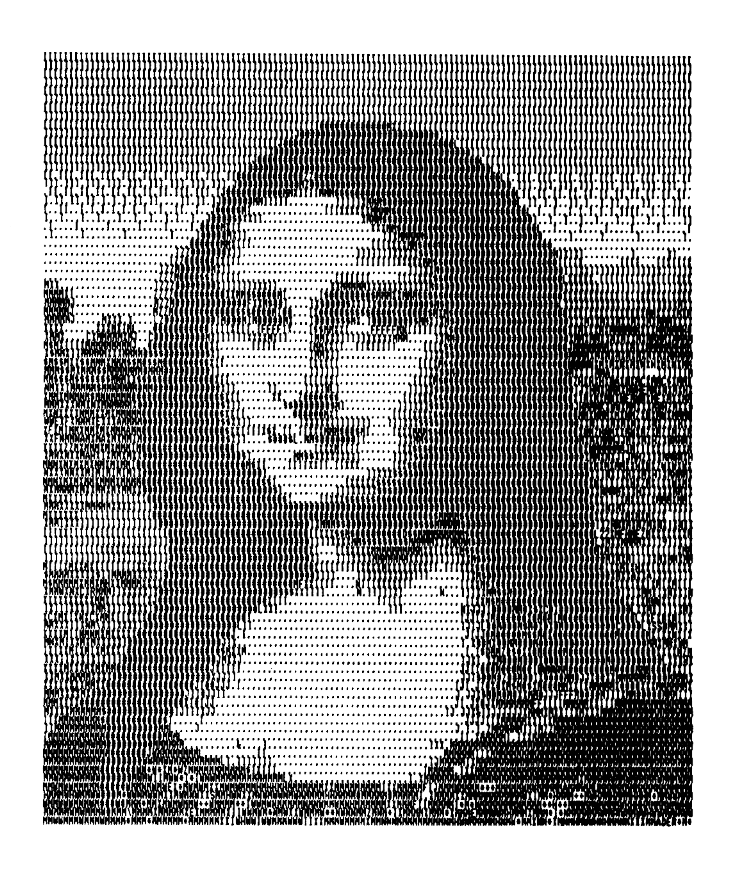
Une juxtaposition astucieuse a permis d'obtenir la simulation d'un dégradé facilitant le 'dessin' d'images qui, vues d'une distance suffisante, sont d'une qualité passable.

Des graphismes de ce type sont trop imprécis quand il s'agit de représentations techniques. Les constructeurs ont donc élargi la police de caractères par un certain nombre de signes spécifiques.

C'était généralement des lignes, angles, arcs de cercle et carrés et rectangles de différents types et dimensions.

Ces signes graphiques annonçaient l'ère des jeux d'ordinateur animés. Et déjà en 1974, on trouvait un nouveau jeux vidéo de ping-pong; ce jeu ne comprenait que des blocs rectangulaires représentant soit la

balle, la raquette ou encore la délimitation du champs de jeu.



graphisme par bloc

Lés principaux jeux d'ordinateur utilisent encore aujourd'hui ce type de graphisme; les caractères graphiques sont en effet nombreux et peuvent s'utiliser sans gros efforts de programmation.

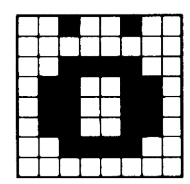
Il est néanmoins toujours ennuyeux de représenter une oblique à 22,5 degrés alors que la police de caractère de l'ordinateur ne dispose que d'une diagonale à 45 degré.
Le programmeur aurait alors été le plus heureux des hommes si il avait eu le possibilité de redéfinir lui même les caractères graphiques. Une possibilité que nous n'avons plus à envisager aujourd'hui, puisque nous pouvons adresser individuellement chaque point.

L'utilisateur de l'Amstrad dispose de nombreuses possibilités; non seulement il peut adresser chaque point et tracer des lignes, utiliser les caractères alphabétiques et graphiques, mais il peut aussi redéfinir toute la police de caractères. La création d'un langage codé ne serait pas un problème.

Même les calculs seront pris en charge par votre ordinateur comme nous allons vous le montrer à l'exemple de la redéfinition du 'é'.

définition des caractères

Prenez tout d'abord un papier quadrillé sur lequel vous marquez un emplacement de 8 carrés sur 8.
Essayez ensuite de dessiner la lettre 'é' ou tout autre caractère dans ce carré.
Pour parfaire l'impression que va vous donner cette lettre à l'écran, noircissez les cases que vous avez choisi pour la création de votre caractère.



Inscrivez ensuite en correspondance à chaque colonne une série de '1' et '0'. Le '1' représentera une case pleine alors que le '0' sera noté pour une case vide.
Les chiffres binaires que vous venez d'inscrire correspondent à la matrice utilisée par l'électronique de votre ordinateur pour allumer les différents points d'un caractère à l'écran. Ce caractère sera affiché par l'intermédiaire du système d'exploitation, après lui en avoir précisé le code décimal.

Que de calculs en perspective alors que chaque étudiant utilise automatiquement sa calculatrice; c'est déjà un argument important à l'encontre d'une redéfinition de caractère.

Et si en plus vous voulez utiliser vos dix doigts pour ne pas vous plonger dans le système binaire...

Le CPC peut heureusement se charger de ces calculs avec l'instruction 'PRINT &X 00111100'.
Le chiffre binaire choisi ci-dessus n'est qu'un exemple, vous pouvez naturellement indiquer n'importe quel autre chiffre binaire.

00000100
00001000
00111100
01100110
01111110
01100000
00111100
00000000

Lorsque vous aurez décidé des 8 chiffres décimaux, vous aurez à définir quel sera le caractère à remplacer par celui que vous venez de créer. Cette opération sera à terminer par l'intruction 'SYMBOL x,w1,w2,w3,w4,w5,w6,w7,w8'. 'x' sera le numéro de code du caractère (CHR$) à modifier et 'w1' à 'w8' reprendra la valeur décimale calculée ci-dessus.
Il ne faudra auparavant pas oublier l'instruction 'SYMBOL AFTER x', ou 'x' désignera un code caractère à sauter lors de la modification d'un de ceux-ci.

Comme nous allons le voir par la suite, les caractères graphiques en particulier et les lettres en général sont d'une importance primordiale. Le programme que nous allons vous présenter ci de suite sera semblable à la cassette de démonstration de l'Amstrad CPC et vous prendra en charge les calculs ainsi que la redéfinition d'un caractère.

utilisation : éditeur de caractères

Le programme se composera de deux parties:

1.Création à l'écran du nouveau caractère

2.Traitement des données et redéfinition

Il serait en effet inconcevable de créer le caractère à la main sur papier pour n'effectuer que les calculs sur ordinateur.
nous devons tout d'abord inscrire une grille de 8 cases sur 8 à l'écran, et ensuite déplacer le curseur à l'intérieur de celle-ci.

L'ordinateur gardera en mémoire si un point est mis ou non le plus facilement par le tableau à deux dimensions:

```
10 DIM caractère$(8,8)
20 FOR ligne=1 TO 8
30 FOR colonne=1 TO 8
40 caractère$(ligne,colone)=CHR$(144)
50 NEXT colonne,ligne
```

Le coin supérieur gauche de l'écran étant positionné en 1,1, la sortie de cette zone sur le moniteur ne posera aucun problème:

```
70 GOSUB 500
500 LOCATE 1,1:FOR li=1 TO 8
510 FOR co=1 TO 8
520 PRINT caractère$(li,co);
530 NEXT co
54o PRINT
550 NEXT li
570 RETURN
```

La représentation à l'écran correspond ainsi exactement à la disposition des données dans le tableau; les positions de lignes et colonnes pourront également être utilisées pour l'adressage du curseur.
Le déplacement du curseur se fera en appuyant sur l'une des touches du pavé numérique; la direction se fera selon la position de cette touche par rapport aux autres. Le numéro de ligne / colonne sera augmenté ou diminué d'un point:

```
80 ligne=1:colonne=1
100 entrée$=INKEY$
110 IF entrée$="8" THEN ligne=ligne-1
120 IF entrée$="2" THEN ligne=ligne+1
130 IF entrée$="6" THEN colonne=colonne+1
140 IF entrée$="4" THEN colonne=colonne-1
150 IF entrée$="7" THEN ligne=ligne-1:
    colonne=colonne-1
160 IF entrée$="9" THEN ligne=ligne-1:
    colonne=colonne+1
170 IF entrée$="3" THEN ligne=ligne+1:
    colonne=colonne+1
180 IF entrée$="1" THEN ligne=ligne+1:
    colonne=colonne-1
```

Vous remarquerez que ce programme gère le déplacement du curseur dans notre éditeur de caractères, mais peut également diriger le déplacement d'un vaisseau spatial si vous le souhaitez.

Il suffit de toujours entrer les coordonneés de l'origine (ici 1,1), qui seront ensuite modifiées en fonction de l'appui sur les touches.
Bien entendu vous pouvez vous référer à une position graphique, un 'SHAPE' ou un caractère avec l'instruction 'TAG', et déplacer

n'importe quel emplacement de l'écran, la vitesse du déplacement étant déterminée par le pas choisi dans l'addition ou la soustraction.

Mais justement pour les jeux, et peut être aussi pour l'application présente, vous souhaiterez contrôler les déplacements non pas par le clavier, mais plutôt par une manette de jeu.

Vous souvenez-vous de notre explication concernant les 'SHAPES' ?

La manette de jeu se comporte de la même façon; elle transmet au calculateur un chiffre définissant exactement une direction ou fonction:

1-haut
2-bas

4-gauche
8-droite

16-touche II
32-touche I

Les valeurs pour des directions composées, les diagonales seront données par l'addition des valeurs des deux directions adjacentes.
Si vous appuyez en outre sur le bouton 'FEU' pendant que vous indiquez une direction, la valeur sera augmentée d'un certain nombre.
L'exploitation en basic se réalise par l'instruction 'JOY(x)', dans laquelle 'x' prendra la valeur 0 pour la manette de jeu 1 et la valeur 1 pour la manette de jeu 2, puis en utilisant de la façon habituelle l'instuction 'INKEY'.

Une partie du programme pourrait avoir l'aspect suivant:

```
80 ligne=1:colonne=1
105 entrée=JOY(0)
```

```
115 IF entrée=1 THEN ligne=ligne-1
125 IF entrée=2 THEN ligne=ligne+1
135 IF entrée=8 THEN colonne=colonne+1
145 IF entrée=4 THEN colonne=colonne-1
155 IF entrée=5 THEN ligne=ligne-1:
    colonne=colonne-1
165 IF entrée=9 THEN ligne=ligne-1:
    colonne=colonne+1
175 IF entrée=10 THEN ligne=ligne+1:
    colonne=colonne+1
185 IF entrée=6 THEN ligne=ligne+1:
    colonne=colonne-1
```

Lors de la création de notre nouveau caractère il sera souhaitable d'imprimer une étoile dans la case du curseur afin d'en connaitre la position.
Celle-ci devra s'effacer après déplacement du curseur, notre matrice de 8 sur 8 serait sinon rapidement remplie d'étoiles.
Cette marque pourrait également clignoter, ce qui ne demande pas de routine spécifique; il suffira de relancer le programme depuis le début:

```
225 LOCATE colonne,ligne:PRINT"*";
270 GOTO 100
```

Le positionnement d'un point sera enregistré par appui sur la touche ENTER, connue par l'ordinateur sous le code CHR$(13). L'affichage à l'écran sera réalisé par le signe graphique 143, un carré plein.
L'enregistrement par erreur d'un point pourra être annulé et enlevé de la matrice par appui sur la touche d'espace:

```
230 IF entrée$=CHR$(13) THEN caractère$
    (ligne,colonne)=CHR$(143)
240 IF entrée$=CHR$(32) THEN caractère$
    (ligne,colonne)=CHR$(144)
250 IF entrée$="D" THEN GOTO 300
```

La ligne 250 permet, après entrée d'un 'D', de redéfinir un nouveau caractère:

```
300 LOCATE 1,1:FOR ligne=1 TO 8
310 FOR colonne=1 TO 8
320 IF caractère$(ligne,colonne)=CHR$(143)
    THEN bit$(ligne)=bit$(ligne)+"1" ELSE
    bit$(ligne)=bit$(ligne)+"0"
330 NEXT colonne
340 PRINT bit$(ligne):bit$(ligne="&X"+bit$
    (ligne):code(ligne)=VAL(bit$(ligne))
360 NEXT ligne
370 INPUT"code caractère à modifier (CHR$)";z
380 SYMBOL z,code(1),code(2),code(3),code(4),
    code(5),code(6),code(7),code(8)
390 LOCATE 9,30:PRINT CHR$(a):GOTO 20
```

Ligne par ligne, la représentation à l'écran sera transposée en chiffres binaires (ligne 320).
La chaîne de bits ainsi obtenue sera transposée en chiffre décimaux puis mise en mémoire dans un tableau. Dès que l'utilisateur aura choisi le caractère à redéfinir, cette redéfinition se fera en ligne 380. Le nouveau caractère sera immédiatement affiché pour contrôle en position écran 9,30.

chaînes

Si vous vous demandez dans quel but nous faisons tout ceci, nous pouvons vous répondre que la technique exposée vous permet de créer très rapidement des dessins tout comme des graphiques annimés ou des jeux vidéo.
Tout comme une gestion de textes ou fichiers peut permettre la construction de mots en partant des différents caractères, vous pouvez également construire des chaines graphiques:

```
10 MODE 1
20 Z$=STRING$(240,238)
30 FOR I%=1 TO 4
40 PRINT Z$;
50 NEXT I%
60 PRINT STRING$(40,238);
70 GOTO 70
```

Il vous sera impossible d'obtenir l'affichage d'un tel graphique plus rapidement !

Naturellement une telle chaîne peut se composer de plusieurs symboles, en fonction de votre utilisation:

```
10 MODE 0
20 M1$=CHR$(248)
30 M2$=CHR$(249)
40 M3$=CHR$(250)
50 M4$=CHR$(251)
60 homme$=M1$+M2$+M3$+M4$
70 CLS
80 PRINT homme$
90 PRINT
```

Le principal avantage de cette chaine graphique est la possibilité, en

plus de toute la palette d'instruction de traitement de chaînes du basic intégré, d'utiliser tout les types d'adressage écran. Vous pouvez positionner la représentation graphique de cette chaîne avec l'instruction 'LOCATE' à un endroit quel qu'il soit de l'écran, tout comme vous pouvez la placer grâce à l'instruction 'TAG' et à l'aide du curseur graphique à l'un des 128.000 emplacements possible.

```
10 MODE 0
20 M1$=CHR$(248)
30 M2$=CHR$(249)
40 M3$=CHR$(250)
50 M4$=CHR$(251)
60 homme$=M1$+M2$+M3$+M4$
70 CLS
80 colonne=1:ligne=1:stylo=1
90 PEN stylo
100 LOCATE colonne,ligne
110 PRINT homme$
120 ligne=ligne+1
130 IF ligne<25 THEN 100 ELSE ligne=1
140 colonne=colonne+4
150 stylo=stylo+1
160 IF stylo=16 THEN stylo=1
170 PEN stylo
180 IF colonne>20 THEN colonne=1
190 GOTO100
```

caractères de contrôle

Les nombreuses possibilités de création de caractères graphiques sont loin d'être épuisées.
Peut-être avez vous déjà remarqué que la valeur indiquée après 'SYMBOL AFTER' est toujours supérieure à 32. Seuls les caractères dont le code est compris entre 32 et 255 peuvent donc être modifiés.
En effet les 32 premiers codes correspondent chacun à une fonction bien déterminée de commande du système. Ceux-ci ne peuvent donc pas être modifiés si le fonctionnement irréprochable du système doit être maintenu.
Cela ne veut pas dire qu'il est impossible d'utiliser ces caractères de contrôle.

Voyez la différence entre le programme suivant et le précédent. Les codes caractères (CHR$) permettent ici de choisir le mode de fonctionnement, d'effacer l'écran tout comme de choisir la couleur:

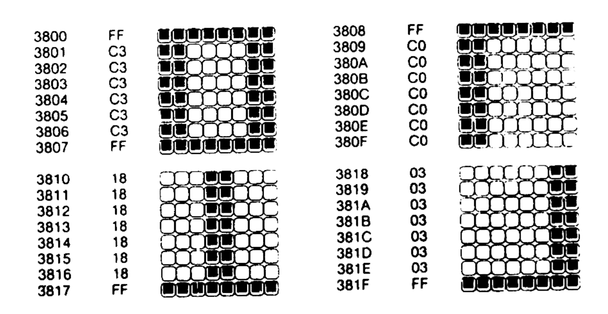
```
10 PRINT CHR$(4);CHR$(0)
20 M1$=CHR$(248)
30 M2$=CHR$(249)
40 M3$=CHR$(250)
50 M4$=CHR$(251)
60 homme$=M1$+M2$+M3$+M4$
70 PRINT CHR$(12)
80 colonne=1:ligne=1:stylo=1
90 PRINT CHR$(15);CHR$(stylo)
100 LOCATE colonne,ligne
110 PRINT homme$
120 ligne=ligne+1
130 IF ligne<25 THEN 100 ELSE ligne=1
140 colonne=colonne+4
```

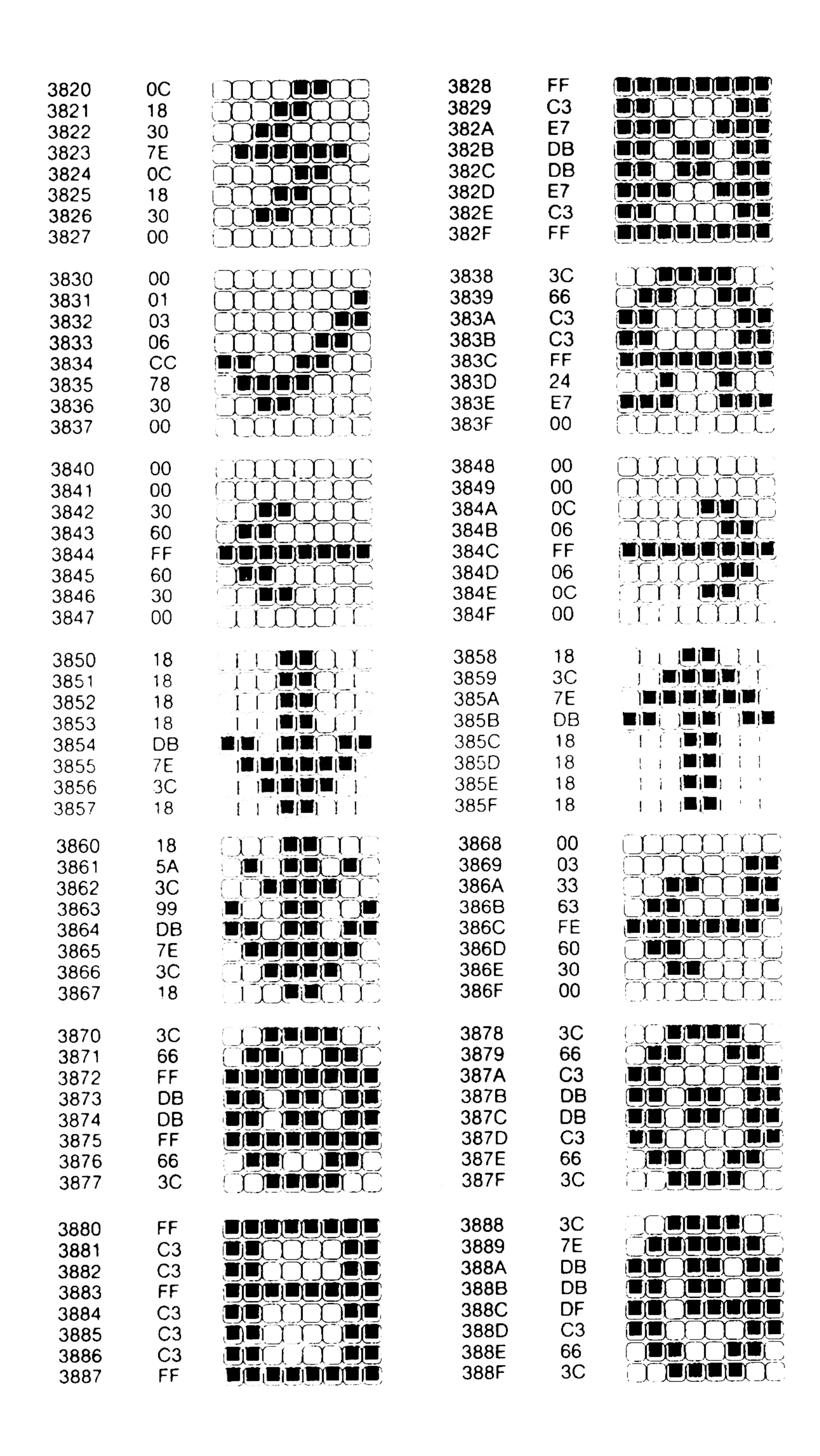
```
150 stylo=stylo+1
160 IF stylo=16 THEN stylo=1
170 PRINT CHR$(15);CHR$(stylo)
180 IF colonne>20 THEN colonne=1
190 GOTO 100
```

Et ce n'est toujours pas tout ! Votre Amstrad peut encore plus. Vous pouvez aussi positionner ainsi votre curseur à n'importe quel endroit de l'écran.

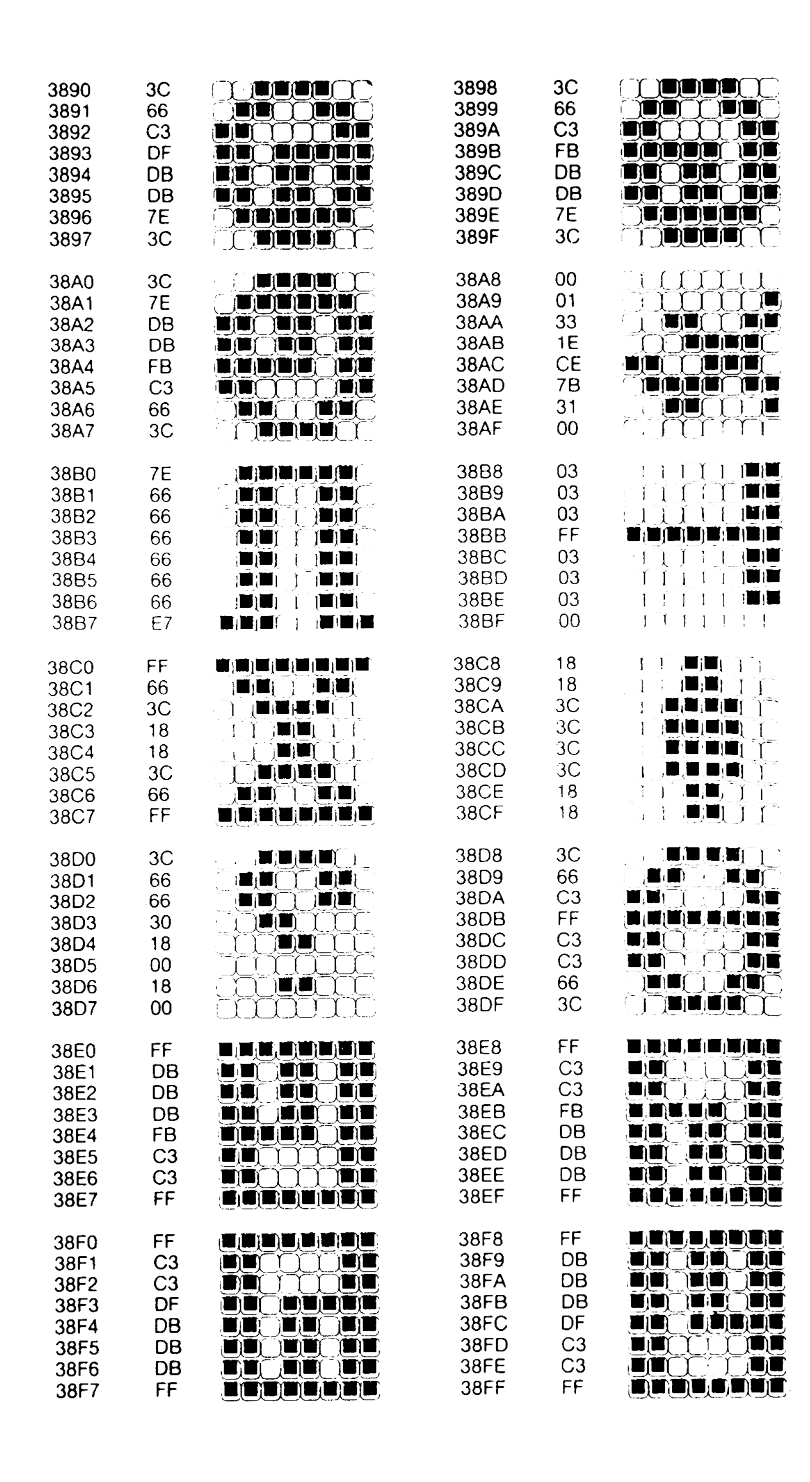
Les pages suivantes vous donnent plus de détails sur les fonctions de chaque caractère de contrôle.
En pratique les 32 premiers caractères ont une fonction double puisque ce sont d'une part des caractères de contrôle qui d'autre part ont eux même une représentation graphique.
Afin d'en obtenir l'affichage, il sera bien entendu nécessaire soit, en mode direct d'appuyer sur la touche contrôle, soit en mode programmation d'indiquer tout d'abord 'PRINT CHR$(2);'.

L'équivalence graphique de ces différents caractères de contrôle n'étant pas reprise dans l'annexe 3 du manuel d'emploi de l'Amstrad, nous en avons repris la représentation ci-dessous.





3820	0C
3821	18
3822	30
3823	7E
3824	0C
3825	18
3826	30
3827	00
3828	FF
3829	C3
382A	E7
382B	DB
382C	DB
382D	E7
382E	C3
382F	FF
3830	00
3831	01
3832	03
3833	06
3834	CC
3835	78
3836	30
3837	00
3838	3C
3839	66
383A	C3
383B	C3
383C	FF
383D	24
383E	E7
383F	00
3840	00
3841	00
3842	30
3843	60
3844	FF
3845	60
3846	30
3847	00
3848	00
3849	00
384A	0C
384B	06
384C	FF
384D	06
384E	0C
384F	00
3850	18
3851	18
3852	18
3853	18
3854	DB
3855	7E
3856	3C
3857	18
3858	18
3859	3C
385A	7E
385B	DB
385C	18
385D	18
385E	18
385F	18
3860	18
3861	5A
3862	3C
3863	99
3864	DB
3865	7E
3866	3C
3867	18
3868	00
3869	03
386A	33
386B	63
386C	FE
386D	60
386E	30
386F	00
3870	3C
3871	66
3872	FF
3873	DB
3874	DB
3875	FF
3876	66
3877	3C
3878	3C
3879	66
387A	C3
387B	DB
387C	DB
387D	C3
387E	66
387F	3C
3880	FF
3881	C3
3882	C3
3883	FF
3884	C3
3885	C3
3886	C3
3887	FF
3888	3C
3889	7E
388A	DB
388B	DB
388C	DF
388D	C3
388E	66
388F	3C



3890	3C
3891	66
3892	C3
3893	DF
3894	DB
3895	DB
3896	7E
3897	3C
38A0	3C
38A1	7E
38A2	DB
38A3	DB
38A4	FB
38A5	C3
38A6	66
38A7	3C
38B0	7E
38B1	66
38B2	66
38B3	66
38B4	66
38B5	66
38B6	66
38B7	E7
38C0	FF
38C1	66
38C2	3C
38C3	18
38C4	18
38C5	3C
38C6	66
38C7	FF
38D0	3C
38D1	66
38D2	66
38D3	30
38D4	18
38D5	00
38D6	18
38D7	00
38E0	FF
38E1	DB
38E2	DB
38E3	DB
38E4	FB
38E5	C3
38E6	C3
38E7	FF
38F0	FF
38F1	C3
38F2	C3
38F3	DF
38F4	DB
38F5	DB
38F6	DB
38F7	FF

3898	3C
3899	66
389A	C3
389B	FB
389C	DB
389D	DB
389E	7E
389F	3C
38A8	00
38A9	01
38AA	33
38AB	1E
38AC	CE
38AD	7B
38AE	31
38AF	00
38B8	03
38B9	03
38BA	03
38BB	FF
38BC	03
38BD	03
38BE	03
38BF	00
38C8	18
38C9	18
38CA	3C
38CB	3C
38CC	3C
38CD	3C
38CE	18
38CF	18
38D8	3C
38D9	66
38DA	C3
38DB	FF
38DC	C3
38DD	C3
38DE	66
38DF	3C
38E8	FF
38E9	C3
38EA	C3
38EB	FB
38EC	DB
38ED	DB
38EE	DB
38EF	FF
38F8	FF
38F9	DB
38FA	DB
38FB	DB
38FC	DF
38FD	C3
38FE	C3
38FF	FF

les caractères de contrôle

Toutes les routines des caractères de contrôle dont le code est compris entre 1 et 31 peuvent être lancées par l'intermédiaire d'une instruction 'PRINT'.

Exemple: PRINT CHR$(7) permettra d'émettre un son.

Des opérations plus complexes demandent d'autres indications. Il faudra dans ce cas ajouter un point virgule suivi des codes CHR$ des paramètres.

Exemple: PRINT CHR$(1);CHR$(1) réalise l'affichage à l'écran d'un carré.

Ces codes sont:

CHR$(0)
non défini

CHR$(1)
Ordonne la sortie du caractère graphique indiqué par le code CHR$ suivant. L'instruction PRINT CHR$(1);CHR$(1) permettra l'affichage d'un carré à l'emplacement du curseur.

CHR$(2)
Le curseur en mode texte ne sera plus affiché.

CHR$(3)
Remet l'affichage du curseur en service.

CHR$(4)
La valeur x comprise dans l'instruction PRINT CHR$(x) définira le mode de fonctionnement du CPC. Le traitement de cette variable se fait en module 4; ce ne sera pas la valeur réelle qui déterminera le mode de

fonctionnement, mais le reste d'une division par 4 de cette variable. PRINT CHR$(4);CHR$(5) sera identique à PRINT CHR$(4); CHR$(1); le mode 1 sera activé

CHR$(5)

Correspond à PRINT CHR$(2), sauf que le caractère sera ici affiché à l'emplacement du curseur graphique.

CHR$(6)

Active l'écran texte.

CHR$(7)

Correspond à la cloche du code ASCII; émet un bip sonore et vide la file d'attente sonore

CHR$(8)

Déplace le curseur d'une position en arrière

CHR$(9)

Déplace le curseur d'une position en avant

CHR$(10)

Déplace le curseur d'une ligne vers le bas

CHR$(11)

Déplace le curseur d'une ligne vers le haut

CHR$(12)

Correspond à l'instruction CLS

CHR$(13)

correspond à l'appui sur la touche ENTER

CHR$(14)

Interprète le code suivant MODULO 16 en tant qu'instruction PAPER.

CHR$(15)

Interprète le code suivant en tant qu'instruction PEN:
PRINT CHR$(15);CHR$(2) correspond à PEN 2
Il sera pris en compte le reste de la division MOD 16.

CHR$(16)
Efface le caractère sous le curseur.

CHR$(17)
Efface la ligne courante jusqu'au curseur

CHR$(18)
Efface la ligne courante du curseur jusqu'à sa fin

CHR$(19)
Efface l'écran de la position 1,1 au curseur

CHR$(20)
Efface l'écran du curseur jusqu'à sa fin

CHR$(21)
Déconnecte l'écran texte

CHR$(22)

Le reste de la division par 2 du code suivant sera utilisé pour activer et débrancher le mode transparent.

PRINT CHR$(22);CHR$(0) débranche le mode transparent alors que PRINT CHR$(22);CHR$(1) l'Active.

CHR$(23)

Le résultat de la valeur suivant MOD 4 décidera du mode d'affichage graphique.

Si vous envoyez après CHR$(22) un CHR$(0) à

l'écran, le mode usuel sera activé; tous les points devant être posés pour l'exécution des prochaines instructions graphiques le seront.

Les autres trois possibilités poseront les nouveaux points à afficher en relation aux coordonnées des points existants.

PRINT CHR$(22);CHR$(1) n'affichera que les points déjà présents et ceux à afficher;
Suivi de CHR$(2), seul les points communs au deux types d'écran seront allumés.

CHR$(24)

Echange les couleurs entre PAPER et PEN.

CHR$(25)

Correspond au symbole de commande. Tout comme en basic, l'instruction PRINT CHR$(25); devra être suivie de 9 paramètres précisant le caractère à modifier ainsi que les points à allumer.

CHR$(26)

Est l'équivalent de l'instruction basic WINDOW. Quatre paramètres définissant les coins de la fenêtre sont à indiquer.

CHR$(28)

Correspond à l'instruction INK; les paramètres suivant sont à préciser: Numéro de stylo, couleur 1, couleur 2

CHR$(29)

Correspond à l'instruction BORDER; couleur 1 et couleur 2 sont à préciser

CHR$(30)

Place le curseur dans le coin supérieur gauche de la fenêtre en cours.

CHR$(31)
Remplace tout à fait l'instruction LOCATE. Indiquer comme d'habitude le numéro de colonne et de ligne.

Certes tous les codes d'instruction ne méritent pas notre attention; certain font se poser la question suivante: Pourquoi faire simple en utilisant les instructions basic alors que l'on peut se compliquer la vie !?
Les codes qui nous intéressent tout particulièrement sont ceux ayant une action sur le curseur, la couleur et le mode de fonctionnement.

Une utilisation astucieuse de ces codes ouvre aux possesseurs de l'Amstrad CPC un nombre de possibilités bien plus important qu'aux fans du Commodore et de ses SPRITES.

Jeux d'action

Les possibilités fantastiques de ceux-ci comptent pour beaucoup de gens comme étant la plus performante des caractéristiques d'un ordinateur. Ces mêmes personnes jugeront leur matériel trop souvent en fonction des jeux disponibles et non pas de la facilité de programmation.

Les jeux vidéo professionnels installés dans les salles de jeux font souvent appel à des micro ordinateurs spécialisés disposant d'extensions spécifiques et d'une progammation particulière. Du côté technique, il pourra s'agir d'un nombre important de SPRITES d'un accès très rapide, ou d'un tube cathodique couleur de très haute définition.

Coté programmation, ces jeux étaient auparavant écrits en langage machine pour garantir le déroulement rapide des opérations.
De nos jours nous utilisons dans ce domaine un langage de programmation évolué afin de réduire le temps de développement de ces programmes et donc d'en limiter le coût.
On utilise en principe le langage FORTH développé pour répondre aux besoins de l'astronomie. Ce langage permet de créer sans restriction ses propres mots d'instruction.
Une version du GAMEFORTH pourrait comprendre les instructions JOYMOVE ou EXPLODE dont l'intégration dans un programme FORTH entrainerait le déplacement d'une figure selon la position de la manette de jeu ou imiterait en son et image une explosion.

Mais les ordinateurs domestiques sont polyvalants. Heureusement les contructeurs ont tenu compte de l'intérêt que porte la clientèle à ces jeux d'action. Chaque ordinateur dispose donc de L'une ou l'autre des fonctions spécifiques à ce type de programmation.

L'Amstrad CPC n'échappe pas à cette règle. Bien qu'il n'ait pas de SPRITES, il bénéficie de nombreuses aides à la programmation de jeux.

Les différentes techniques de création et d'animations graphiques sont développées dans les pages suivantes.

graphisme - chaînes de caractères

Ces chaînes peuvent être créées à volonté. Seule la place mémoire disponible pourrait freiner les élans créatifs d'un programmeur.
Et si vous arrivez à saturation de la mémoire, votre programme débordera de toute façon de fusées, vaisseaux, monstres, hommes ou autres.

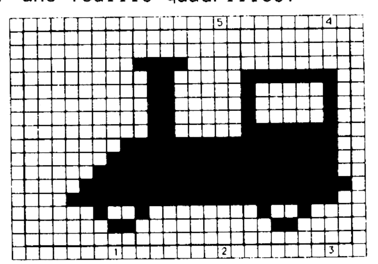
Chaque figure devant être bien visible et comprendre des détails, la taille devra être proportionnée.
Choisir le mode 0 et redéfinir un caractère devrait suffire à première vue. Mais les inconvénients suivants sautent rapidement aux yeux:

- les déplacements se font par saccades.

- chaque figure ne peut avoir qu'une seule couleur.

Il n'y a donc plus que le mode 1 qui peut être envisagé: la définition est relativement bonne et nous disposons encore des 4 couleurs.
Les figures devront se composer de plusieurs caractères, tout comme nous l'avons déjà fait pour nos 'hommes'.

Si nous avions par exemple besoin d'une locomotive, il faudrait tout d'abord définir sa taille approximative - deux caractères sur trois - puis devrions en tracer les contours sur une feuille quadrillée:



Nous remplissons ensuite l'intérieur de la surface selon notre idée puis redéfinissons un nombre correspondant de caractères.
Il y en aurait ici cinq dont les nouvelles données donneraient:

1,3,7,15,31,4,3,0
255,255,255,255,255,128,0,0
254,254,254,255,254,72,48,0
0,0,0,254,130,130,130,254
0,0,240,96,96,96,96,96

Ces valeurs peuvent être soit calculées, soit données par notre éditeur de caractères.
Mais cela ne sert pas à grand chose de modifier une fois les caractères. Ce dont nous avons besoin est en fait un sous-programme comprenant les diverses instructions BASIC recquises.
Et pourquoi cette routine ne serait-elle pas intégrée à notre éditeur de caractères?

Une fois ce programme complèté, l'ordinateur nous déchargera de tout ce travail.
Toutes les valeurs des différents caractères à modifier seront mémorisées dans une zone spécifique. Une fois la création terminée, toutes les données seront transférées y compris l'instruction SYMBOL sur notre support magnétique.
Ce fichier pourra ensuite être rechargé en mémoire comme tout autre programme, ou bien ajouté à un programme en mémoire avec l'instruction MERGE.

Nous avons pour cette raison repris ci-dessous le listing intégral de notre éditeur de caractères ZPRO.

```
5 MODE 1:PAPER 0:INK 0,12:BORDER 12:PEN 1:INK 1,0:SYMBOL AFTER 33
10 DEFINT a-z:DIM caractère$(8,8):nb=0
20 CLS:FOR ligne=1 TO 8
30 FOR colonne=1 TO 8
```

```
40 caractère$(ligne,colonne)=chr$(144)
50 NEXT colonne
60 bit$(ligne)=""
70 NEXT ligne
80 GOSUB 400:GOSUB 500
90 ligne=1:colonne=1
100 entrée$=INKEY$:entrée=JOY(0)
110 IF entrée$="8" THEN ligne=ligne-1
115 IF entrée=1 THEN ligne=ligne-1
120 IF entrée$="2" THEN ligne=ligne+1
125 IF entrée=2 THEN ligne=ligne+1
130 IF entrée$="6" THEN colonne=colonne+1
135 IF entrée=8 THEN colonne=colonne+1
140 IF entrée$="4" THEN colonne=colonne-1
145 IF entrée=4 THEN colonne=colonne-1
150 IF entrée$="7" THEN ligne=ligne-1: colonne=colonne-1
155 IF entrée=5 THEN ligne=ligne-1: colonne=colonne-1
160 IF entrée$="9" THEN ligne=ligne-1: colonne=colonne+1
165 IF entrée=9 THEN ligne=ligne-1: colonne=colonne+1
170 IF entrée$="3" THEN ligne=ligne+1: colonne=colonne+1
175 IF entrée=10 THEN ligne=ligne+1: colonne=colonne+1
180 IF entrée$="1" THEN ligne=ligne+1: colonne=colonne-1
185 IF entrée=6 THEN ligne=ligne+1: colonne=colonne-1
190 IF ligne<1 THEN ligne=1: PRINT CHR$(7)
200 IF ligne>8 THEN ligne=8: PRINT CHR$(7)
210 IF colonne<1 THEN colonne=1: PRINT CHR$(7)
220 IF colonne>8 THEN colonne=8: PRINT CHR$(7)
225 LOCATE colonne,ligne:PRINT"*";
230 IF entrée$=CHR$(13) THEN caractère$(ligne,colonne)= CHR$(143)
235 IF entrée=17 THEN caractère$(ligne,colonne)= CHR$(143)
240 IF entrée$=CHR$(32) THEN caractère$(ligne,colonne)= CHR$(144)
```

```
245 IF entrée=18 THEN caractère$(ligne,colonne)= CHR$(144)
250 IF entrée$="P" THEN GOTO 300
251 IF entrée$="D" THEN GOTO 600
255 LOCATE 9,8:PRINT ligne,colonne
260 GOSUB 500
270 GOTO 100
300 LOCATE 1,1:FOR ligne=1 TO 8
310 FOR colonne=1 TO 8
320 IF caractère$(ligne,colonne)= CHR$(143) THEN bit$(ligne)
=bit$(ligne)+"1" ELSE bit$(ligne)= bit$(ligne)+"0"
330 NEXT colonne
340 PRINT bit$(ligne): bit$(ligne)="&x"+bit$(ligne): code (ligne)= VAL
(bit$(ligne))
360 NEXT ligne
370 INPUT "quel caractère modifier (code chr$)";z
380 SYMBOL z, code(1), code(2), code(3), code(4), code(5), code(6),
code(7), code(8)
385 nb=nb+1: zalt(nb)=z: FOR i=1 TO 8: zcode(nb,i)= code(i): NEXT i
390 LOCATE 9,30:PRINT CHR$(a): GOTO 20
400 WINDOW#1,10,40,1,10
410 PRINT#1,"* * * *    Z - PRO    * * * *"
420 PRINT#1: PRINT#1,"mouvement curseur:  pavé  numérique":PRINT#1,"ou
manette de jeux"
430 PRINT#1,"mettre un point: ENTER"
440 PRINT#1,"redéfinition: SHIFT P"
450 RETURN
499 REM affichage de la matrice
500 LOCATE 1,1: FOR re=1 TO 8
510 FOR sp=1 TO 8
520 PRINT caractère$(re,sp);
530 NEXT sp
```

```
540 PRINT
550 NEXT re
570 RETURN
600 WINDOW#2,1,40,10,20:CLS 2
610 PRINT#2,"préparer support magnétique !"
620 PRINT#2,"cr=ation du fichier de DATA:"
630 INPUT#2,"nom ";file$: IF LEN(file$)>8 THEN 630
640 INPUT#2,"premier numéro de ligne ";zn
650 INPUT#2,"pas ";sw
660 file$="!"+file$
670 OPENOUT file$
680 PRINT#9,STR$(zn)+ "SYMBOL AFTER 32"
690 FOR z=1 TO nb
700 PRINT#9,STR$(zn)+ "SYMBOL "+ STR$(zalt(z))+ ","+ STR$(zcode(z,1))+
","+ STR$(zcode(z,2))+ ","+ STR$(zcode(z,3))+  ","+  STR$(zcode(z,4))+
","+ STR$(zcode(z,5))+ ","+ STR$(zcode(z,6))+  ","+  STR$(zcode(z,7))+
","+ STR$(zcode(z,8))
710 zn=zn+sw
720 NEXT z
730 CLOSEOUT
740 RUN
```

Pour les besoins de notre exemple, nous avons modifié les caractères 130,131,132,133 et 134 afin qu'ils définissent chaque élément de notre locomotive dans le sens inverse de celui des aiguilles d'une montre.

Pour afficher le résultat à l'écran, nous utilisons ces codes ainsi que les codes de déplacement du curseur puisque notre dessin est à cheval sur deux lignes.

```
10 REM train1
20 MODE 0
30 SYMBOL AFTER 129
40 SYMBOL 130,1,3,7,15,31,4,3,0
50 SYMBOL 131,255,255,255,255,255,128,0,0
60 SYMBOL 132,254,254,254,255,254,72,48,0
70 SYMBOL 133,0,0,0,254,130,130,130,254
80 SYMBOL 134,0,0,240,96,96,96,96,96
90 LOCATE 10,10: REM où?
100 PRINT CHR$(130);: REM nez du train
110 PRINT CHR$(131);: REM chaudière
120 PRINT CHR$(132);: REM arrière de la locomotive
130 PRINT CHR$(11);: REM monter d'une ligne
140 PRINT STRING$(2,8);: REM deux cases vers la gauche
150 PRINT CHR$(134);: REM cheminée
160 PRINT CHR$(133);: REM poste de conduite
170 LOCATE 1,1
```

Si cette locomotive doit se déplacer à l'écran, l'affichage sous forme de caractères est trop lourd à employer et trop lent.
Heureusement que le BASIC de l'AMSTRAD accepte des chaines de caractères pouvant contenir jusqu'à 255 caractères. Ceci devrait être suffisant dans tous les cas de figures.

```
180 train$= CHR$(130)+ CHR$(131)+ CHR$(132)+  CHR$(11)+  STRING$(2,8)+
```

```
CHR$(134)+ CHR$(133)
190 LOCATE 3,3
200 PRINT train$
210 LOCATXE 1,1
```

De cette façon, vous pouvez maintenant, avant un programme de jeux, redéfinir les caractères et intégrer ces éléments dans le déroulement du jeu.

Il est préférable d'utiliser des désignations claires dans votre programme. Celui-ci sera facilement compréhensible. Il n'en serait pas de même si le programmeur n'indique que des chiffres ou noms de variables sans signification qu'il aurait d'ailleurs oublié au bout de quelques heures.

représentations multi-couleurs

Elles sont très simples à réaliser en partant de ce que nous avons dit plus haut. Nous avons déjà mentionné le code 15 qui remplace l'instruction PEN.

Les modifications suivantes permettent d'obtenir un corps de locomotive noir et une cabine bleu ainsi que la cheminée.

Vous remarquerez qu'il s'agit bien de l'instruction PEN et non INK. Il ne faut donc pas oublier de préciser en début de programme les couleurs en même temps que le type de mode de fonctionnement.

```
10 REM train1
20 MODE 0: INK 1,1: INK 2,0
30 SYMBOL AFTER 129
40 SYMBOL 130,1,3,7,15,31,4,3,0
50 SYMBOL 131,255,255,255,255,255,128,0,0
60 SYMBOL 132,254,254,254,255,254,72,48,0
```

```
70 SYMBOL 133,0,0,0,254,130,130,130,254
80 SYMBOL 134,0,0,240,96,96,96,96,96
90 LOCATE 10,10: REM où?
100 PRINT CHR$(130);: REM nez du train
110 PRINT CHR$(131);: REM chaudière
120 PRINT CHR$(132);: REM arrière de la locomotive
130 PRINT CHR$(11);: REM monter d'une ligne
140 PRINT STRING$(2,8);: REM deux cases vers la gauche
150 PRINT CHR$(15); CHR$(2);: REM maintenant stylo 2
160 PRINT CHR$(134);: REM cheminée
170 PRINT CHR$(133);: REM poste de conduite
180 PRINT CHR$(15); CHR$(1);: REM et retours au stylo 1
190 LOCATE 1,1
```

Ceci peut aussi être inclus dans la chaine graphique:

```
200 stylo1$= CHR$(15)+ CHR$(1)
210 stylo2$= CHR$(15)+ CHR$(2)
220 train$= CHR$(130)+ CHR$(131)+ CHR$(132)+  CHR$(11)+  STRING$(2,8)+
stylo2$+ CHR$(134)+ stylo1$+ CHR$(133)
230 LOCATE 4,4: PRINT train$
```

Toutes les instructions de traitement de chaînes de caractères pourront être utilisées pour manier ces chaînes graphiques. Vous pouvez donc n'utiliser qu'une partie du dessin, ou en interchanger d'autres, toutes les possibilités vous sont ouvertes!

Animation graphique

Lors de la projection d'un filme, l'impression de mouvement provient du nombre d'images projetées à une vitesse supérieure à ce que peut discerner l'oeil humain.
Si l'on regarde de près les 25 images TV, ou les 18 images d'un filme 8 mm, qui défilent dans les deux cas en une seconde, on constate qu'un mouvement d'une seconde est en fait divisé en 25 ou 18 mouvements intermédiaires.
Nous voyons donc une suite d'images de mouvements saccadés, mais dont l'oeil ne peut discerner les nuances dès que la cadence est supérieure à 14 images par secondes.

La grande aventure du film est la conséquence de l'inertie de l'oeil; mais l'oeil ne se laisse pas toujours tromper.
Si la différence entre les différentes positions des images est trop grande, le mouvement se fera par saccades même si la fréquence de passage des images est élevée.
Ce phénomène se retrouve dans les films, en particulier si un objet ou une personne traverse l'écran en droite ligne d'un bout à l'autre.
C'est bien pour cela que ce type de plan est évité!
Le metteur en scène préfèrera dans une poursuite, que l'automobile surgisse de l'arrière plan, change de direction puis disparaisse dans le fond.

Nous utiliserons également cette technique de défilement d'images. Si nous pouvions persévérer dans cette optique d'animation de dessins par ordinateur, nous obtiendrions sur notre moniteur un dessin animé de très bonne qualité.

Mais ceci suposerait qu'au moins seize fois par secondes, l'écran serait reconstitué. Cet énorme travail d'affichage de points (16 * 640 * 200) serait impossible à réaliser, même en langage machine.

Nous n'avons pas besoin de générer à chaque fois l'image entière, mais uniquement la partie mobile.

Nous allons afficher un point, l'effacer, calculer sa nouvelle position, le réafficher, l'effacer une nouvelle fois, recalculer sa nouvelle position...

```
200 FOR zeile=1 TO 24
210 FOR colonne=1 TO 40
220 LOCATE colonne,zeile
230 PRINT " *";
240 NEXT colonne
250 NEXT zeile
```

Après avoir lancé ce programme par l'instruction RUN 200, l'étoile "*" se déplacera sur l'écran du coin en haut à gauche, à celui en bas à droite.
Puisque nous affichons d'abord un blanc, ensuite seulement notre caractère, et que la nouvelle position n'avance que d'une position, l'ancien affichage sera automatiquement effacé.

En marche arrière, cela fonctionne aussi bien comme le montre un essai après avoir modifié PRINT "* " ainsi que les boucles qui décomptent jusqu'à un.
Cette façon de faire pose certains problèmes, en particulier au bords de l'écran, lors de l'adressage de la prochaine ligne.

Une astuce peut être la mémorisation de la dernière position affichée; l'ancienne image peut ainsi être effacée avant d'afficher la suivante:

```
199 salt=1:zalt=1
200 FOR zeile=1 TO 24
```

```
210 FOR colonne=1 TO 40
220 LOCATE salt,zalt
230 PRINT " ";
240 LOCATXE colonne,zeile
250 PRINT "*";
260 salt=colonne:zalt=zeile
270 NEXT colonne
280 NEXT zeile
```

N'oubliez pas d'indiquer manuellement les premières coordonnées du point à effacer, comme ici en ligne 199.
Sinon, ce programme fonctionne bien; il peut néanmoins encore être amélioré.

Le CPC dispose d'une routine système attendant l'affichage du dernier point sur le tube cathodique; un écran n'est affiché sur le moniteur qu'une fois entièrement constitué.
Cette routine peut être appelée par le vecteur BD19; naturellement aussi par le BASIC.

Complétons notre programme:

```
245 CALL &BD19
```

Notre étoile traverse maintenant parfaitement bien l'écran.
Nous pouvons aussi prendre notre train ou tout autre objet comme le montre l'exemple suivant:

```
10 REM train2
20 MODE 0: DEFINT a-z
30 SYMBOL AFTER 129
40 SYMBOL 130,1,3,7,15,31,4,3,0
50 SYMBOL 131,255,255,255,255,255,128,0,0
```

```
60 SYMBOL 132,254,254,254,255,254,72,48,0
70 SYMBOL 133,0,0,0,254,130,130,130,254
80 SYMBOL 134,0,0,240,96,96,96,96,96
90 zug$= CHR$(130)+ CHR$(131)+ CHR$(132)+ CHR$(11)+ STRING$(2,8)+
CHR$(134)+ CHR$(133)
100 effacetrain$="   "+ CHR$(11)+ STRING$(2,8)+ "  "
110 ORIGIN 0,47:DRAW 640,0
120 salt=73: zalt=22
130 zeile=22
140 PRINT CHR$(2)
150 FOR colonne=73 TO 1 STEP -1
160 LOCATE salt,zalt
170 CALL &BD19
180 PRINT effacetrain$;
190 LOCATXE colonne,zeile
200 CALL &BD19
210 PRINT zug$;
220 salt=colonne
230 NEXT colonne
240 salt=1: GOTO 150
```

L'intérêt de ce listing réside dans la définition d'une chaine de caractères, dans laquelle les blancs sont disposés de façon identique aux différents composants du dessin.
Il est bien évident qu'un seul blanc ne peut effacer notre chaine train$.

Le prochain pas consistera à abandonner l'instruction de positionnement LOCATE pour placer notre chaine à l'aide du curseur graphique à n'importe quel endroit de notre écran de 640 * 200 points.

Au lieu de LOCATE, nous utiliserons MOVE, il faudra aussi adapter la longueur des boucles:
Un caractère représente huit points sur huit; un pas de 10 devrait être adapté à notre déplacement.

Il y a quand même un inconvénient; les instructions de déplacement du curseur n'affectent pas le curseur graphique.
Notre locomotive à vapeur se transforme ici en autorail moderne.

```
10 REM train3
20 MODE 2: DEFINT a-z
30 SYMBOL AFTER 129
40 SYMBOL 130,1,3,7,15,31,4,3,0
50 SYMBOL 131,255,255,255,255,255,128,0,0
60 SYMBOL 132,254,254,254,255,254,72,48,0
70 MODE 2
80 ORIGIN 1,1
90 PLOT 1,35: DRAW 640,35
100 a$=CHR$(130)+ CHR$(131)+ CHR$(132)
110 TAG
120 FOR x=601 TO 1 STEP -10
130 MOVE xalt,50
140 CALL &BD19
```

```
150 PRINT "    ";
160 MOVE x,50
170 CALL &BD19
180 PRINT a$;
190 xalt=x
200 NEXT x
210 GOTO 120
```

Pourquoi des SPRITES ?

Chaque utilisateur d'un AMSTRAD CPC se réservera cette question pour les personnes croyant disposer avec la poignée de SPRITES de leur ordinateur du nec plus ultra de la programmation graphique.
Il est certain que les SPRITES sont simples à diriger, d'autre part, il ne détruisent pas le fond sur lequel ils se déplacent; mais qui a dit que les chaines que nous utilisons effacent le fond?
Les concepteurs du CPC ont en effet envisagé ce cas de figure. Mais l'utilisation de cette possibilité demande un peu plus d'expérience et de connaissances que le manuel d'emploi n'en dispense.

Le mot magique est CHR$(23). Notre train peut maintenant passer devant n'importe quelle gare sans devoir la redessiner après.

Mais comment cela fonctionne-t-il?

Eh bien, c'est très simple; selon les besoins, une relation logique sera établie entre les anciens et les nouveaux points, que ce soit avec OR, AND, ou encore XOR.

Malheureusement, ces termes ne sont pas très connus des programmeurs BASIC. Ceux-ci peuvent prèsque toujours contourner ces termes complexes, à l'encontre des virtuoses du langage machine.
OR, XOR ou AND sont utilisés pour formuler des conditions complexes, qui peuvent souvent être remplacées par plusieurs conditions IF, et aussi pour positionner, effacer ou définir un bit dans un octet.

La meilleure manière pour en comprendre la logique est de se constituer une table de vérité.

La condition ET n'est remplie que si les deux implications sont positives:

0 AND 0 > 0

0 AND 1 > 0
1 AND 0 > 0
1 AND 1 > 1

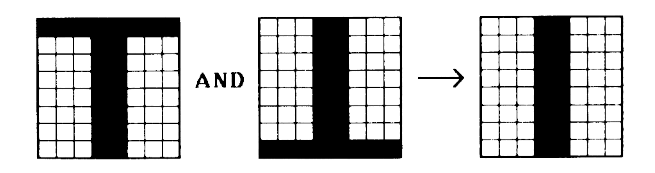
La condition OU n'est vraie que si l'une des deux valeurs est positive:

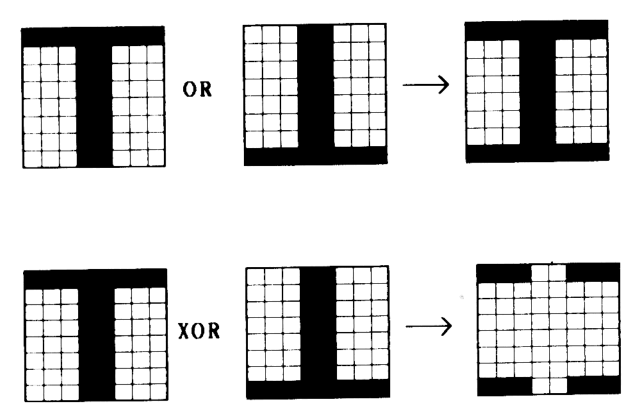
0 OR 0 > 0
0 OR 1 > 1
1 OR 0 > 1
1 OR 1 > 1

XOR est un 'OU exclusif', ce qui signifie qu'une valeur au minimum ne doit pas être vraie, mais peut l'être.

0 XOR 0 > 0
0 XOR 1 > 1
1 XOR 0 > 1
1 XOR 1 > 1

Maintenant, si vous voulez vous habituer aux effets obtenus par PRINT CHR$(23);CHR$(1) etc... imaginez vous que l'opérande gauche corresponde à l'arrière plan de l'écran, l'opérande droite à la forme à déplacer et le 1 à un point allumé.
Ce qui restera après un PRINT sera identique au chiffre placé à droite du signe .





Une explication de ces liaisons peut aussi être donnée à l'aide des lignes suivantes:

```
10 MODE 0
20 LOCAT E 1,1
30 INPUT "0,1,2 ou 3 pour *";E
40 PRINT CHR$(23);CHR$(E)
50 TAG
60 MOVE 100,100
70 PRINT "*";
80 TAGOFF
90 LOCATE 1,2
100 INPUT"0,1,2 ou 3 pour%";E1
110 PRINT CHR$(23);CHR$(E1)
120 TAG
130 MOVE 100,100
140 PRINT "%";
150 TAGOFF
160 GOTO 20
```

Pour notre programme, nous devrions utiliser le 1, la relation OR.

La première instruction PRINT affiche tous les points présents dans les deux images. Si l'un des points est ensuite affiché une nouvelle fois exactement au même endroit, le résultat de la condition OR sera obligatoirement différent.

Notre locomotive pourra passer devant une gare, bien que celle-ci ne soit ici représentée que d'une façon symbolique.

```
10 REM train4
20 REM "demo SPRITE"
30 REM -------------
40 MODE 2: DEFINT a-z
50 SYMBOL AFTER 129
60 SYMBOL 130,1,3,7,15,31,4,3,0
70 SYMBOL 131,255,255,255,255,255,128,0,0
80 SYMBOL 132,254,254,254,255,254,72,48,0
90 MODE 2
100 ORIGIN 1,1
110 PLOT 1,35: DRAW 640,35
120 MOVE 100,50: TAG: PRINT"fond";
130 train$=CHR$(130)+ CHR$(131)+ CHR$(132)
140 TAVGOFF
150 PRINT CHR$(23); CHR$(1)
160 TAG
170 For x=701 TO -29 STEP -10
180 MOVE x,50
190 CALL &BD19
200 PRINT train$;
210 MOVE x+10,50
220 PRINT train$;
230 NEXT x
```

```
240 GOTO 170
```

Utilisation: jeux d'arcade

Le programme suivant utilise encore une fois certaines des routines que nous venons de voir.
Il simule un simple jeu de tir, du style 'tir aux canards' ou 'ball-trap'; le nom exact dépendra de celui qui le baptisera!

Le but de ce jeu est d'abattre le plus grand nombre de cibles se déplaçant sur l'écran de gauche à droite.
Le joueur dispose d'un certain nombre de balles qu'il pourra tirer d'un point mobile de l'écran.
Pour motiver le joueur, le nombre de points marqués ainsi que la quantité de munitions restantes sont affichés pendant la durée du jeu.

Il s'agit en quelque sorte d'un programme standard pouvant être la base de votre propre programme.

Tous les pas importants de la programmation d'un jeu s'y trouvent:
Le joueur se dirigera avec les touches 'Z' et ' ', mais il pourrait diriger comme le curseur de notre éditeur de caractères, dans plus de deux directions par exemple un vaisseau spatial.
Pourquoi n'y aurait-il pas des envahisseurs qui descendraient sur l'écran et un tir de fusées pour leur barrer la route?
Et même la routine de score peut être incorporée au jeu, la rapidité de celui-ci dépendant par exemple du score. Quel entrain!

```
10 DEFINT r,s
20 DEFREAL g,z
30 MODE 1
40 s=20: zs=1: rz=1: tirmax=20
50 GOSUB 300
60 e$=INKEY$
70 sbalt=s
```

```
80 LOCATE zs,rz: PRINT" ";: REM effacer cible
90 IF cible=0 THEN rz=RND(1) * 20 + 0.5: projectile=RND (1) * 2:
cible=-1: zs=1: REM nouvelle cible
100 zs= zs+ projectile: IF zs>38 THEN cible0
110 CALL &BD19
120 LOCATE zs,rz: PRINT CHR$(225);: REM cible
130 IF e$="z" THEN s=s-1: IF s<1 THEN s=1
140 IF e$="\" THEN s=s+1: IF s>40 THEN s=40
150 IF (e$=" " AND tir=0) THEN tir=-1: ss=s: rs=22: comptetir=
comptetir+1: GOSUB 300
160 IF tir=-1 THEN GOSUB 220
180 LOCATE sbalt,23: PRINT" ";: REM effacer base
190 LOCATE s,23: PRINT"^";: REM base
200 GOTO 60
210 REM tir
220 LOCATE ss,rs: PRINT" ";: REM effacer tir
230 IF (rs=rz AND ss=INT(zs)) THEN punkt= INT(punkt+100*projectile):
tir=0: cible=0: GOSUB 300: PRINT CHR$(7): GOTO 260
240 rs=rs-1: IF rs=0 THEN tir=0: GOTO 250 ELSE LOCATE ss,rs: PRINT
CHR$(239);: REM tir
250 IF (tir=0 AND comptetir= tirmax) THEN GOTO 270
260 RETURN
270 LOCAT E 9,10: PRINT"* * * GAME OVER * * *";
280 e$=INKEY$: IF e$="" THEN 280
290 IF e$= CHR$(13) THEN RUN ELSE 280
300 LOCATE 1,24: PRINT"SCORE: "; tirmax - comptetir;
310 RETURN
```

Les variables sont initialisées en ligne 40, au début du jeu. Elles correspondent à:

s = colonne écran de la base

s = colonne écran de la base
zs = colonne écran de la cible
rz = ligne écran de la cible

Ensuite, la routine d'affichage du score en ligne 300 est appelée. La ligne 60 décèle une action du joueur. Si c'est le cas, son ancienne position est notée pour garantir son effacement par la suite.

En ligne 90 est contrôlé si il y a encore une cible à l'écran. Si il n'y en a plus, ligne et vitesse du projectile seront aléatoires. Après calcul des nouvelles coordonnées, et un test évitant de dépasser les valeurs limites des coordonnées de l'écran, la cible est affichée. Maintenant est traitée la touche auparavant enfoncée - le joueur voulait-il se déplacer à gauche ou à droite, ou encore voulait-il tirer une balle à l'aide de la touche d'espace? - Dans ce cas, les différents compteurs sont mis à jour et un drapeau est mis, permettant au sous-programme de déplacement du projectile de se dérouler si nécessaire uniquement.

Les deux autres points à réaliser par le programme principal sont l'effaçage et l'affichage de la position du joueur.

Utilisation: éditeur graphique

Avant de continuer ce livre et d'entrer dans les formules mathématiques et la technique de programmation graphique en deux et trois dimensions, nous vous présentons un autre programme qui pourra vous être tout comme le générateur de caractères très utile par la suite.

Il s'agit d'un éditeur graphique, d'un programme vous permettant d'utiliser le curseur et le moniteur comme un crayon sur une feuille de papier.
Mais nous n'en restons pas là; outre une simple fonction de dessin, cet éditeur prendra en charge la programmation et génèrera par appui sur une touche des programmes BASIC complets pouvant être par la suite ajoutés à vos propres programmes à l'aide de MERGE, comme nous l'avons déjà fait précédemment avec SYMBOL.

Le coté pratique de ce programme vous sera évident au plus tard quand vous aurez programmé un jeu à l'aide des techniques précédentes et que vous souhaitez y ajouter un décor.
Vous aurez remarqué qu'en essayant de reproduire un dessin du papier à l'écran dans un sous-programme, les lignes parallèles sont plus ou moins inclinées; c'est parce qu'il ne vous est pas possible d'adresser le point exact parmi les 128.000 possibles.

Ce programme aura pour principale tâche de tenir à jour toutes les coordonnées.
C'est pour cela que nous prévoyons plusieurs tableaux pour contenir les variables x et y de chaque point (60).
L'éditeur doit aussi être en mesure de savoir si un point est bien seulement un point ou si c'est peut-être la fin d'une droite; c'est le rôle de la variable modus$. Xm et ym sont des variables auxiliaires utilisées seulement pour le tracé de cercles.

L'affichage d'un point tout comme le tracé d'une ligne ne demande pas une grande programmation et peut donc être éxécuté immédiatement après l'ordre correspondant (530,540).
Les autres routines prenant en charge le tracé de cercles, la programmation d'un dessin ainsi que la sauvegarde et le chargement en mémoire d'un écran graphique sont des sous-programmes.

Leur construction devrait être assez compréhensible, les variables étant connues et les désignations n'étant pas abréviées.

Il y aurait peut-être la ligne 6030 qui requiert quelques explications.
Ici, le curseur est déplacé pendant la routine de tracé d'un cercle (GOSUB 7010). Ensuite est calculé le rayon en fonction de la distance des coordonnées x et y du point par rapport au centre du cercle. Nous avons déjà vu cette procédure dans les premières pages de ce livre.

```
10 REM **********************************
20 REM * éditeur graphique; version 1.0 *
30 REM **********************************
40 MODE 1: PAPER 0: PEN 1: INK 0,12: BORDER 12: INK 1,0
50 DEFINT a-z: ORIGIN 1,1
60 DIM x(200), y(200), modus$(200), xm(200), ym(200)
70 c=10: couleur=1
80 REM PEN 1: INK 1,12
90 LOCATE 7,5: PRINT "éditeur de graphiques et programmes"
100 LOCAT E 13,6: PRINT"par Joerg Walkowiak"
110 LOCATE 1,12: PRINT STRING$(40,154)
120 LOCATE 10,20: PRINT CHR$(164);" 1985 - MICRO APPLICATIONS"
130 FOR i=1 TO 5000: NEXT
140 GOTO 1010
500 REM ************* programme principal
510 entrée$=INKEY$
520 GOSUB 7010: REM déplacer curseur
530 IF entrée$="p" THEN x(p)=x: y(p)=y: modus$(p)="p": PLOT
x,y,couleur: p=p+1: lx=x: ly=y: GOTO 510: REM *** affichage point
540 IF entrée$="l" THEN modus$(p)="l": x(p)=x: y(p)=y: PLOT
x,y,couleur: DRAW x,y,couleur: lx=x: ly=y: p=p+1: GOTO 510: REM ***
tracer ligne
550 IF entrée$="P" THEN GOTO 2010
560 IF entrée$="z" THEN GOTO 5010
570 IF entrée$="S" THEN GOTO 3010
550 IF entrée$="L" THEN GOTO 4010
590 IF (entrée$="h" OR entrée$="H") THEN GOTO 1010
600 IF entrée$="c" THEN CLS: INPUT "nouveau pas de déplacement du
curseur";c: GOTO 5010
610 IF entrée$="@" THEN p=p-1: GOTO 5010
620 IF entrée$="k" THEN 6000: REM dessin cercle
```

```
630 GOSUB 8010: REM clignotement curseur
640 GOTO 510
650 REM************* fin du programme principal
1000 REM *********** instructions
1010 MODE 1: CLS: LOCATE 15,1: PRINT " menue d'aide": PRINT
STRING$(40,154);
1020 PRINT"7 8 9          ";: PEN 2: PRINT"déplacement curseur": PEN 1
1030 PRINT"4   6          en fonction de la"
1040 PRINT"1 2 3          position des 12 touches
1045 PRINT"               du pavé numérique
1050 PRINT"  5            au milieu de l'écran": PRINT
1060 PEN 2: PRINT"commandes";: PEN 1: PRINT " en minuscules:"
1070 PRINT" p - afficher un point au curseur"
1080 PRINT" l - tracer ligne du dernier point"
1090 PRINT" k - centre du cercle. ENTER après"
1095 PRINT"     choix du rayon"
1100 PRINT" z - redessiner page écran"
1110 PRINT" c - choix du pas de déplacement curseur"
1120 PRINT
1130 PEN 2: PRINT"commandes";: PEN 1: PRINT " en majuscules:"
1140 PRINT" H - menu d'aide"
1150 PRINT" S - sauvegarde données écran"
1160 PRINT" L - charger données écran"
1170 PRINT" P - générer programme BASIC"
1180 PEN 2: PRINT"@ - annule dernière instruction": PEN 1
1190 PRINT STRING$(40,154);: PRINT"appuyer sur une touche";
1200 IF INKEY$="" THEN 1200
1210 CLS: x=320: y=200: GOTO 510
2000 REM *********programmation
2010 CLS: PRINT"programmation graphique...": PRINT STRING$(40,154)
2020 INPUT"nom du programme";nom$: IF LEN(nom$) 16 THEN GOTO 2010
```

```
2030 IF nom$="@" THEN GOTO 5010
2040 PRINT: INPUT"no. de la première ligne du PRG"; ligne
2050 PRINT: INPUT"pas souhaité"; sautligne: PRINT
2060 WINDOW#1,1,40,18,25
2070 OPENOUT nom$
2080 ligne$= "ORIGIN 1:1:DEG:GOTO "+ STR$(ligne+ 2* sautligne):
PRINT#1, ligne; ligne$: PRINT#9, ligne;ligne$: ligne= ligne+ sautligne
2090 ligne$= "FOR degré=1 TO 360: PLOT r* COS(degré), r* SIN(degré):
NEXT degré: ORIGIN 1,1: RETURN":upcercle=ligne: PRINT 1, ligne;
ligne$: PRINT#9, ligne;ligne$: ligne= ligne+ sautligne
2100 ligne$=""
2110 FOR i=0 TO p-1
2120 IF modus$(i)="p" THEN commande$="plot": GOTO 2150
2130 IF modus$(i)="l" THEN commande$="draw": GOTO 2150
2140 IF modus$(i)="k" THEN ligne$="r="+ STR$(x(i))+ ": ORIGIN "+
STR$(xm(i))+ ","+ STR$(ym(i))+": GOSUB "+ STR$(upcercle): GOTO 2160
2150 ligne$= commande$+ STR$(x(i))+ ","+ STR$(y(i))
2160 IF LEN(progligne$)<2 THEN progligne$= ligne$: ligne$="": GOTO
2200
2170 IF (LEN(progligne$)<200 AND LEN(progligne$)>5)THEN progligne$=
progligne$+ ":"+ligne$:ligne$="": GOTO 2200
2180 PEN 3: PRINT#9, ligne; progligne$: PEN 2: PRINT 1, ligne;
progligne$
2190 ligne = ligne+ sautligne:progligne$= ligne$
2200 NEXT i
2210 IF LEN(progligne$)>1 THEN PEN 3: PRINT#9, ligne; progligne$: PEN
2: PRINT#1, ligne; progligne$
2220 PEN 3: CLOSEOUT
2230 PEN 1
2240 PRINT: INPUT"terminer "; entrée$: IF UPPER$(LEFT$(entrée$,1))="O"
THEN MODE 1: END
```

```
2250 GOTO 5010
3000 REM********** save
3010 CLS: PRINT"sauvegarde données écran...": PRINT STRING$(40,"_")
3020 INPUT"nom";nom$: IF LEN(nom$)>16 THEN GOTO 3010
3030 IF nom$="@" THEN GOTO 5010
3040 OPENOUT nom$
3050 PEN 3: PRINT
3060 PRINT#9, p
3070 FOR i=0 TO p
3080 PRINT#9,x(i),y(i),modus$(i),xm(i),ym(i)
3090 NEXT i
3100 CLOSEOUT
3110 PEN 1
3120 PRINT: INPUT"terminer "; entrée$: IF UPPER$(LEFT$(entrée$,1))="O"
THEN MODE 1: END
3130 GOTO 5010
4000 REM ********* chargement
4010 CLS: PRINT"chargement des données écran...": PRINT
STRING$(40,"_")
4020 INPUT"nom";nom$: IF LEN(nom$) 16 THEN GOTO 4010
4030 IF nom$="@" THEN GOTO 5010
4040 PRINT: PEN 2
4050 OPENIN nom$
4060 INPUT#9, p
4070 FOR i=0 TO p-1
4080 INPUT#9,x(i),y(i),modus$(i),xm(i),ym(i)
4090 NEXT i
4100 CLOSEIN
4110 PEN 1
5000 REM ********* dessin
5010 ORIGIN 1,1: CLS: FOR i=0 TO p-1
```

```
5020 IF modus$(i)="p" THEN PLOT x(i),y(i),couleur: GOTO 5040
5030 IF modus$(i)="l" THEN DRAW x(i),y(i),couleur
5040 IF modus$(i)="k" THEN ORIGIN xm(i), ym(i): DEG: rayon= x(i):  FOR
degré=1 TO 360: PLOT rayon* COS(degré),  rayon*  SIN(degré),  couleur:
NEXT degré: ORIGIN 1,1
5050 NEXT i
5060 GOTO 510
6000 REM ********* cercle
6010 xm=x: ym=y: ORIGIN xm,ym: x=0: y=0: DEG
6020 entrée$= INKEY$: GOSUB 8010: IF entrée$="" THEN 6020
6030 IF entrée$<>CHR$(13) THEN GOSUB 7010: rayon= SQR(x*x+y*y): GOSUB
8010: GOTO 6020
6040 FOR degré=1 TO 360
6050 x= rayon* COS(degré):y= rayon* SIN(degré):PLOT x,y,couleur
6070 modus$(p)="k":x(p)=rayon: y(p)= rayon: xm(p)=xm: ym(p)=ym: p=p+1
6080 ORIGIN 1,1: x=xm: y=ym: GOTO 510
7000 REM ********* mouvement curseur
7010 IF entrée$="8" THEN Y=Y+C: GOTO 7100
7020 IF entrée$="6" THEN X=X+C: GOTO 7100
7030 IF entrée$="4" THEN X=X-C: GOTO 7100
7040 IF entrée$="2" THEN Y=Y-C: GOTO 7100
7050 IF entrée$="9" THEN x=x+c: y=y+c: GOTO 7100
7060 IF entrée$="1" THEN x=x-c: y=y-c: GOTO 7100
7070 IF entrée$="3" THEN x=x+c: y=y-c: GOTO 7100
7080 IF entrée$="5" THEN x=320: y=200: GOTO:7100
7090 IF entrée$="7" THEN x=x-c: y=y+c: GOTO 7100
7100 RETURN
8000 REM ********* clignotement curseur
8010 couleur1= TEST(x,y): REM clignotement curseur graphique
8020 FOR x1=1 TO 10
8030 PLOT x,y,couleur1+1
```

```
8040 PLOT x,y,couleur1
8050 NEXT x1
8060 RETURN
```

Mode d'emploi de l'éditeur graphique

Toutes les commandes se font par simple appui sur une touche; aucune instruction ne doit être validée par la touche ENTER.

Il y a trois types d'instructions:

1 instructions curseur
Le déplacement du curseur se fait par l'intermédiaire du pavé numérique placé à côté du magnétophone.
La direction est fonction de l'emplacement de la touche, c'est à dire '6' pour la droite, '7' pour la gauche et le haut etc...

Une fonction spécifique est réservée à la touche 5 qui place le curseur au centre de l'écran, en 320 - 200.

2 commandes
Elles sont données par les caractères en majuscules; n'oubliez donc pas d'appuyer sur SHIFT lors de leur entrée.

Les commandes permettent de sauvegarder et charger les écrans graphiques, générer un programme`BASIC ou de redessiner une image pour correction.

3 instructions de dessin
Elles sont éxécutées immédiatement après entrée du caractère minuscule correspondant.

Vous avez aussi toujours accès au menu d'aide, en appuyant soit sur 'h', soit sur 'H'. Toutes les instructions et commandes dont vous disposez y sont énumérées.

POINTS

Pour afficher un point, placez le curseur en position et appuyez sur

'p'.
Le point est tout de suite affiché et ses coordonnées mises en mémoire pour une utilisation ultérieure.

LIGNES

Mettez le curseur au point de départ de votre ligne, appuyez sur'p'. Placez le ensuite à l'autre bout de votre ligne et appuyez sur 'l'.

Une ligne sera toujours tracée en partant du dernier point, que celui-ci ait été placé par 'p', ou qu'il s'agisse de la fin d'une ligne précédente.

CERCLES

Placez le curseur tout d'abord au centre et appuyez sur la touche 'k'. Il faut ensuite définir le rayon; la direction n'a aucune importance. Appuyez une seconde fois sur 'k' et le dessin se réalise.

ERREURS

Vous avez commis une erreur de dessin. Inutile de tout recommencer; appuyez sur l'arabesque (@), l'écran s'efface et se redessine,hormis la dernière instruction.
Vous pouvez répèter cette opération jusqu'à la disparition du dernier point sur l'écran.

SAUVEGARDE ET CHARGEMENT D'UN ECRAN GRAPHIQUE

Appuyez sur 'S' ou 'L' et indiquez un nom d'une longueur maximum de 8 caractères.
Validez avec la touche ENTER, votre instruction est en cours d'éxécution.

GENERER UN PROGRAMME BASIC

Appuyez sur 'P' et indiquez un nom de 8 caractères au plus.
Vous devez aussi indiquer le numéro de la première ligne du programme à générer, et le pas définissant les prochains numéros.
Ensuite sera sauvegardé un programme autonome que vous pourrez utiliser avec MERGE, LOAD ou RUN.
Tenez compte que l'éditeur ne définit pas le mode de fonctionnement; les programmes ainsi générés sont en effet prévus en tant que sous-routines.
L'instruction CLS n'apparait donc pas en début du programme, à vous de la rajouter si vous souhaitez uniquement obtenir l'écran graphique.

Chapitre 4

GRAPHISME EN 2D

techniques

Nous nous sommes contentés jusqu'ici de présenter les instructions graphiques de base de notre AMSTRAD CPC et nous sommes entrainés à faire les différentes manipulations d'écran.

Nous avons vu comment choisir la couleur souhaitée, afficher à l'écran des éléments graphiques, les superposer et dessiner des formes simples.

Il est évident qu'un graphique peut être réalisé seulement en affichant des points, mais la mise au point d'un motif, d'une image avec ces seules instructions serait fastidieuse.

De plus, une image crée de telle sorte ne pourait être traitée par aucun programme devant en modifier l'aspect.

Il est donc nécessaire de faire appel à d'autres moyens que nous trouvons dans les mathématiques.
Certaines formules pourront être considérées comme de véritables 'potions magiques' auxquelles il suffira d'indiquer seulement les valeurs entrant en ligne de compte.

Ce chapitre doit présenter quelques unes des formules de base et les expliquer à l'aide d'exemples.

SHAPES

Ces figures sont présentes sur presque tous les ordinateurs domestiques, dont notre CPC AMSTRAD.
Bien que le BASIC ne comporte aucune instruction y étant relative, et que l'utilisateur ne trouve nulle trace de ce mot dans le manuel d'emploi, nous pouvons en utiliser les propriétés par l'intermédiaire de l'instruction PLOT.

Quelle est la définition du 'SHAPE'? C'est une figure dont les contours sont reliés par une ligne allant de coins en coins.

La description de la maison ci-dessous peut être faite comme suit:

250,180

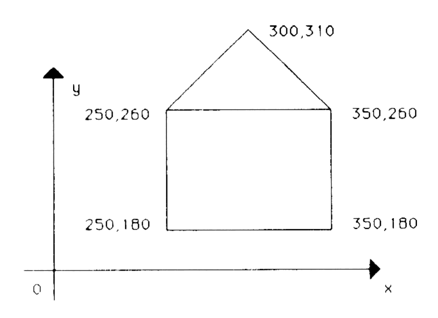
250,260

300,310

350,260

350,180

250,180



Pour en faire l'affichage à l'écran, sur la majorité des ordinateurs, le programme suivant sera employé:

```
10 MODE 1: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
20 PLOT 250,180: REM coté bas, gauche
30 PLOT 250,260: REM coté haut, gauche
40 PLOT 300,310: REM faît du toit
50 PLOT 350,260: REM coté haut, droit
60 PLOT 350,180: REM coté bas, droit
70 PLOT 250,180: REM base de la maison
```

Une autre solution consiste à conserver les données sur une ligne. Les lignes ne seront donc plus tracées directement, mais par l'intermédiaire d'un petit programme prévu à cet effet:

```
10 MODE 1: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
20 READ X,Y: REM premier point
30 PLOT X,Y
40 READ X,Y: REM lire les autres points
50 DRAW X,Y: REM tracer les lignes
60 GOTO 40: REM continuer

1000 REM données de la maison
1020 DATA 250,180,250,260,300,310,350,260
1030 DATA 350,180,250,180
```

Mais le programme s'arrête par un rapport d'erreur parce qu'il ne connait pas le nombre de coins. Il est facile d'y remédier, d'autant plus qu'il s'agit de figures définies dans le cas des SHAPES.

```
10 MODE 1: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
15 READ NB: REM nombre de coins
20 READ X,Y: REM premier point
30 PLOT X,Y
35 FOR POINT=1 TO NB-1
40 READ X,Y
50 DRAW X,Y
80 NEXT POINT
90 END

1000 REM données de la maison
1010 DATA 6
1020 DATA 250,180,250,260,300,310,350,260
1030 DATA 350,180,250,180
```

Le premier élément de données précise le nombre de coins à relier et dont les coordonnées sont dans les lignes 1020 et 1030.

Ce type de programmation vous semble peut-être lourd à employer pour tracer des formes simples; mais voyez comme il est facile de modifier les figures:

```
10 MODE 1: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
15 READ NB: REM nombre de coins
20 READ X,Y: REM premier point
30 PLOT X,Y
35 FOR POINT=1 TO NB-1
40 READ X,Y
50 DRAW X,Y
80 NEXT POINT
90 END

1000 REM nouvelles données du SHAPE
1010 DATA 11,320,240,330,230,330,180,340
1020 DATA 170,330,170,320,180,310,170,300
1030 DATA 170,310,180,310,230,320,240
```

Ce programme, légèrement adapté, est aussi en mesure de créer un nombre indéfini de SHAPES et donc d'images complètes.

Nous devons convenir d'un repère définissant la fin des données.

Dans notre cas, l'origine de nos coordonnées étant 0,0, nous pouvons choisir une valeur négative. Sur des systèmes différents, il faudra prendre une valeur se trouvant en dehors du système polaire.

```
10 MODE 1: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
15 READ NB: REM nombre de coins
20 READ X,Y: REM premier point
25 IF X<0 THEN END
30 PLOT X,Y
```

```
35 FOR POINT=1 TO NB-1
40 READ X,Y
50 DRAW X,Y
80 NEXT POINT
90 GOTO 15

1000 REM données de la maison
1010 DATA 6
1020 DATA 250,180,250,260,300,310,350,260
1030 DATA 350,180,250,180
1040 REM toit
1050 DATA 2,250,260,350,260
1060 REM fenêtre
1070 DATA 5,340,250,320,250,320,230,340
1080 DATA 230,340,250
1090 DATA 5,260,250,260,230,280,230,280
1100 DATA 250,260,250
1110 REM porte
1120 DATA 4,290,180,290,210,310,210,310
1130 DATA 180, 1, -1, 0
```

modifications des coordonnées

Modifiez maintenant le programme précédent comme suit:

```
13 FOR FACTEUR=0.1 TO 1.9 STEP 0.2
14 RESTORE
19 IF NB =0 THEN NEXT FACTEUR
40 PLOT X*FACTEUR, Y*FACTEUR
50 DRAW X*FACTEUR, Y*FACTEUR
```

Vous obtenez ainsi toute une série de maisons, chacunes plus grandes les unes que les autres.
La mise à l'échelle, permettant d'agrandir et de diminuer une représentation, s'obtient par multiplication avec un facteur constant.
Du point de vue mathématique, il s'agit en fait d'une transformation des coordonnées pouvant être schématisée par une matrice.
Ceci ouvre à l'utilisateur un nouvel horizon sur des possibilités que nous allons étudier ci-de suite.

Le système polaire

Déplacements, diminutions et agrandissements s'obtiennent en deux dimensions sans grands efforts de calculs dans un repère cartésien.
Pour bien observer les différentes manipulations que nous allons faire, nous ne pouvons plus nous contenter de travailler uniquement sur le premier cadran.

Pour cette raison, nous transfèrerons dans les programmes suivants l'origine de notre repère cartésien au centre de l'écran. Pour nous faciliter la tâche, nous allons également tracer les axes x et y.

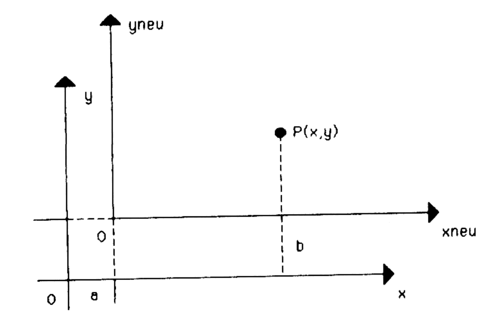
```
10 MODE 2: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
20 ORIGIN 320,200
```

```
100 TAG
110 PLOT -320,0: DRAW 320,0
120 MOVER -5,6: PRINT CHR$(246);
130 MOVER -17,-10: PRINT "x";
140 PLOT 0,-200: DRAW 0,200
150 MOVER -4,-2: PRINT CHR$(244);
160 MOVER -17,-10: PRINT "y";
170 TAGOFF
```

Les axes sont tracés dans les lignes 110 et 140. Les lignes 120 et 150 permettent de placer en retrait le curseur graphique, puis d'afficher les flêches. Il en est de même dans les lignes 130 et 160, pour indiquer l'axe.
Cette particularité, pouvoir écrire avec l'instruction classique PRINT en utilisant les coordonnées graphiques, est activée par l'instruction TAG. TAGOFF permet de revenir à tout moment en mode normal.

Déplacements

Le déplacement parallèle est le plus simple à réaliser dans un repère cartésien.
Il consiste en un mouvement des axes sans en modifier la direction.



Ce schéma montre bien que le déplacement sera fonction des facteurs d'addition a et b.
La relation entre l'ancien et le nouveau système polaire se formule de la façon suivante:

xnouveau = x - a
ynouveau = y - b

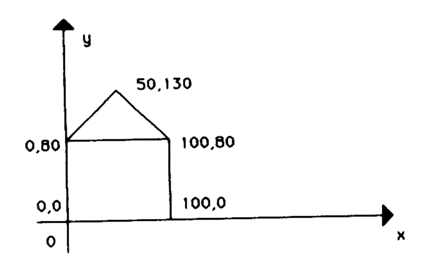
Nous ne voulons pas déplacer les axes, mais le point, ce qui nous donne la formule:

xnouveau = x + a
ynouveau = y + b

Ces deux formules vont nous permettre de transférer notre point en n'importe quel autre endroit de l'écran. Ceci sur un plan, si nous voulons obtenir une perspective, un troisième axe sera à calculer:

znouveau = z + c

Notre maison ci-dessous va nous permettre de démontrer l'exactitude de ce que nous avons développé. Il faut naturellement tenir compte de la modification de l'origine de notre système polaire.

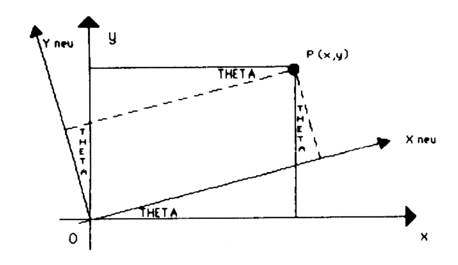


```
5 REM déplacement parallèle
10 MODE 2: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
20 ORIGIN 320,200
30 WINDOW#1,1,35,20,25
99 REM axes
100 CLG: TAG
110 PLOT -320,0: DRAW 320,0
120 MOVER -5,6: PRINT CHR$(246);
130 MOVER -17,-10: PRINT"x";
140 PLOT 0,-200: DRAW 0,200
150 MOVER -4,-2: PRINT CHR$(244);
160 MOVER -17,-10: PRINT"y";
170 TAGOFF
180 IF w=5 THEN 500
199 REM lecture des points
200 RESTORE: READ nb
210 FOR point=1 TO nb
220 READ x(point): READ y(point)
230 NEXT point
240 IF w=1 THEN GOSUB 900
500 CLS#1
510 PRINT#1,"1 - dessin original"
520 PRINT#1,"2 - déplacement en x"
530 PRINT#1,"3 - déplacement en y"
540 PRINT#1,"4 - déplacement en x et y"
550 PRINT#1,"5 - effacer l'écran"
590 INPUT#1,"votre choix";w
600 ON w GOTO 100,700,750,700,100
690 REM déplacement en x
700 INPUT#1,"facteur x ";a
710 FOR point=1 TO nb
```

```
720 x(point)= x(point) + a
730 NEXT point
735 IF w=4 THEN 750
740 GOSUB 900: GOTO 500
750 INPUT#1,"facteur y ";b
760 FOR point=1 TO nb
770 y(point)= y(point) + b
780 NEXT point
790 GOSUB 900: GOTO 500
800 CLG: GOTO 500
899 REM dessin
900 PLOT x(1),y(1)
910 FOR point=1 TO nb
920 DRAW x(point), y(point)
930 NEXT
940 RETURN
999 REM données du dessin
1000 DATA 8,0,0,0,80,050,130,100,80,100,0
1010 DATA 0,0,0,80,100,80
```

Rotations

Elles sont pratiquement aussi simples à réaliser. Nous supposons un mouvement en un plan autour du point d'origine 0,0; l'angle est donné par THETA.



Si vous avez bien retenu le procédé pour obtenir un cercle, vous aurez rapidement fait la liaison entre les points x et xnouveau, y et ynouveau:

xnouveau = x*cos(theta) + y*sin(theta)
ynouveau = -x*sin(theta) + y*cos(theta)

Matrices

Les mouvements de rotation et déplacement vu ci-dessus sont des transformations linéaires.

Elles sont souvent représentées sous forme de matrices, des tableaux de chiffres, en ligne et colonne.
Chaque chiffre de la matrice peut être adressé directement; l'indiçage est similaire à celui que nous obtenons lors du dimensionnement d'une zone, mais sans virgule.

$$M = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}$$

Cette matrice n'est qu'un pur tableau de chiffres contenant uniquement des éléments unitaires, aucune valeur calculée.
D'astucieux mathématiciens n'ont pas pu se tenir tranquilles avant d'avoir rempli des étagères entières de bibliothèques universitaires avec des livres reprenant une multitude de formules de calcul pour tableaux.
Nous ne pouvons aborder ce sujet sans dépasser le cadre de cet ouvrage. Nous ne voulons pas faire un cours de mathématiques, mais bien en savoir plus sur la programmation graphique.
Pour notre utilisation, seuls les additions et multiplications des valeurs de tableaux sont d'une importance. L'exemple ci-dessous doit être suffisant pour en expliquer le fonctionnement.

Considérons les deux matrices M1 et M2:

$$M1= \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \qquad M2= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Pour une addition (soustraction), il suffit d'additionner (soustraire) les valeurs d'indices de ligne et colonne identiques. Le résultat serait:

$$\begin{bmatrix} -1 & 1 \\ 2 & 1 \end{bmatrix}$$

La multiplication semble être assez particulière. Au bout de quelques temps de pratique, il n'y aura plus de problème:
on multiplie la valeur d'une ligne de la première matrice avec la valeur d'une colonne de la seconde. Les produits sont ensuite additionnés:

```
-1 * 0 + 0 * 1          -1 * 1 + 0 * 1

 1 *-1 + 0 * 1           1 * 1 + 0 * 1


               0    -1

              -1     1
```

En règle générale, nous pouvons donner la formule suivante:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} A*a+B*c & A*b+B*d \\ C*a+D*c & C*b+D*d \end{bmatrix}$$

Une modification comme la précédente, un déplacement parallèle peut être schématisé sous la forme du produit de deux matrices, bien qu'il s'agisse d'une multiplication entre une matrice à deux positions et une autre à quatre positions:

$$x \quad y \quad * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a*x+c*y \qquad b*x+d*y$$

x et y représentent les coordonnées du point initial; la somme sur la

gauche les nouvelles valeurs xnouveau et y nouveau.

Le type de modification est défini par les valeurs de la deuxième matrice; le programme suivant permet de bien s'en rendre compte.

```
5 REM démo matrice
10 MODE 2: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
20 ORIGIN 320,200
30 WINDOW#1,1,35,20,25
99 REM axes
100 CLG: TAG
110 PLOT -320,0: DRAW 320,0
120 MOVER -5,6: PRINT CHR$(246);
130 MOVER-17,-10: PRINT"x";
140 PLOT 0,-200: DRAW 0,200
150 MOVER -4,-2: PRINT CHR$(244);
160 MOVER-17,-10: PRINT"y";
170 TAGOFF
199 REM lire points
200 RESTORE: REAXD nb
210 FOR point=1 TO nb
220 READ x(point): READ y(point)
230 NEXT point
500 CLS#1
510 INPUT#1,"a ";a
520 INPUT#1,"b ";b
530 INPUT#1,"c ";c
540 INPUT#1,"d ";d
550 CLS#1
560 PRINT#1,"matrice:"
570 PRINT#1, a,b
580 PRINT#1, c,d
590 INPUT#1,"dessiner ?";e$
600 IF LOWER$(LEFT$(e$,1))= "o" THEN 500
610 CLG: TAG
620 FOR point=1 TO nb
```

```
630 xnouveau(point)= a*x(point)+ c*y(poin)
640 ynouveau(point)= b*x(point)+ d*y(poin)
650 NEXT point
660 PLOT xnouveau(point), ynouveau(point)
670 FOR point=1 TO nb
680 DRAW xnouveau(point), ynouveau(point)
690 NEXT point
700 PLOT -320,0: DRAW 320,0
710 MOVER -5,6: PRINT CHR$(246);
720 MOVER-17,-10: PRINT"x";
730 PLOT 0,-200: DRAW 0,200
740 MOVER -4,-2: PRINT CHR$(244);
750 MOVER-17,-10: PRINT"y";
760 TAGOFF: GOTO 500
770 REM dessin
900 PLOT x(1),y(1)
910 FOR point=1 TO nb
920 DRAW x(point), y(point)
930 NEXT
940 RETURN
1000 DATA 8,0,0,0,80,50,130,100,80,100,0
1010 DATA 0,0,0,80,100,80
```

Une fois ce programme correctement entré et lancé, les données du contour de notre maison sont lues et mises en mémoire dans les variables x et y (ligne 200 à 230).
Une fenêtre, ouverte dans le quatrième cadran permet l'introduction des coordonnées a,b,c et d demandées par le programme.
Ensuite sera affiché le schéma de la matrice. Si vous êtes d'accord, et que vous ne vous êtes pas trompé, vous pouvez confirmer en appuyant sur la touche ENTER pour que le dessin se fasse.

mise à l'échelle

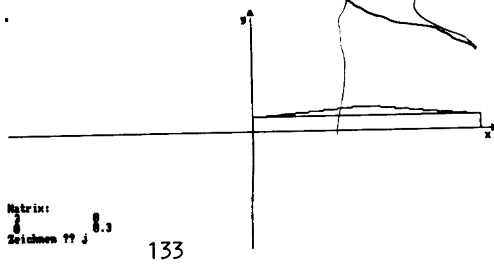
Nous avions constaté que la multiplication de toutes les valeurs de x par un facteur constant agrandit l'axe x. Il en va de même pour l'axe y et ses valeurs.
La multiplication par un chiffre inférieur à 1 entraine une réduction.

Regardons notre matrice en tenant compte de ce nouvel aspect. Nous constatons que ces facteurs sont identiques à a et d (mettez b=c=0).

$$x \quad y \quad * \begin{bmatrix} a & 0 \\ & \\ 0 & d \end{bmatrix} \quad = \quad a*x+0*y,0*x+d*y \; = \; a*x+b*y$$

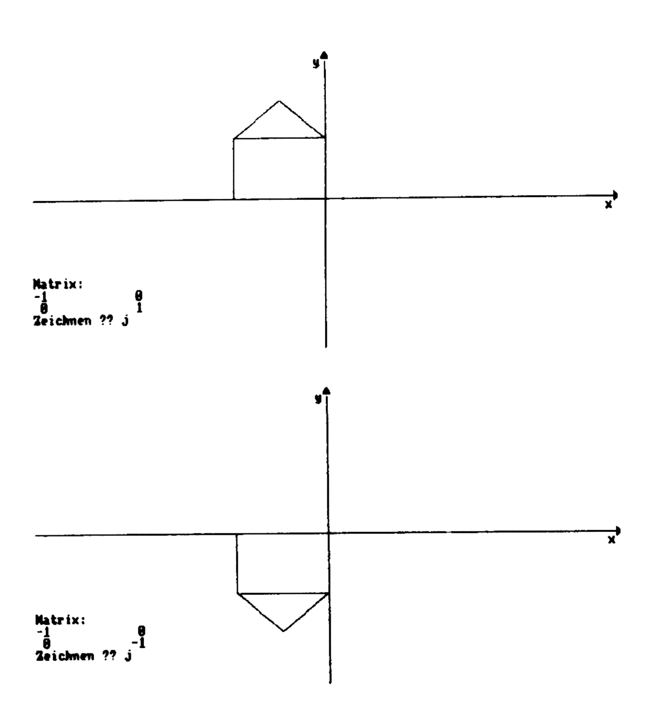
Les valeurs a et d fixent donc la taille du dessin; a définissant un agrandissement ou une réduction sur l'axe x et d sur y.
Des valeurs comprises entre 0 et 1 correspondent à une réduction alors que supérieures à 1, à un agrandissement.

Le dessin ci-dessous montre ainsi une maison de hauteur 3 fois réduite et de largeur triplée.



Une matrice neutre peut également être définie quand les deux facteurs sont égaux à 1 et que le dessin serait donc identique à l'original.

Le résultat d'une telle multiplication de matrices sera une image inversée par rapport à l'axe x, le long de l'axe y.



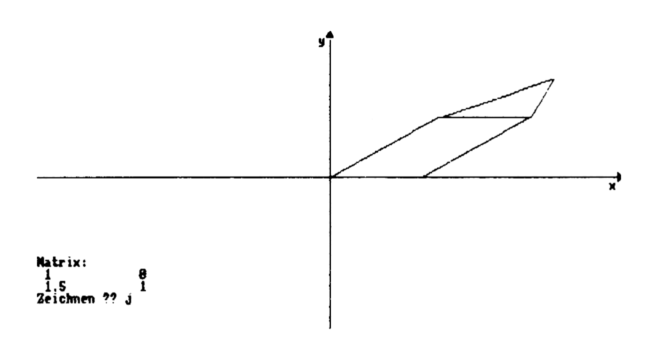
Ces essais deviennent intéressants quand vous indiquez d'autres valeurs b et c. Il en résultera presqu'une impression de perspective; le dessin projeté à l'écran ressemble à l'ombre du dessin de la maison.

Voici le calcul des coordonnées:

$$x \quad y \quad * \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} = x, b*x+y$$

a et d ayant tous deux la valeur 1, aucune mise à l'échelle ne sera faite; l'élément b de notre matrice décalera néanmoins les points sur y au fur et à mesure de l'augmentation de x.
Il en sera de même pour l'axe x si la valeur c est différente de 0.

$$x \quad y \quad * \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = a*x+0*y, 0*x+d*y = a*x+b*y$$



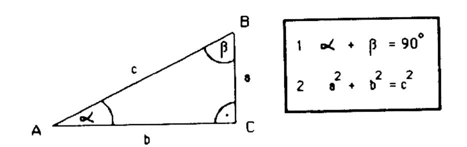
rotations

Le type de mouvement de base qui manque encore à notre panoplie est la rotation d'une figure sur un axe, voir autour de l'origine du système polaire.

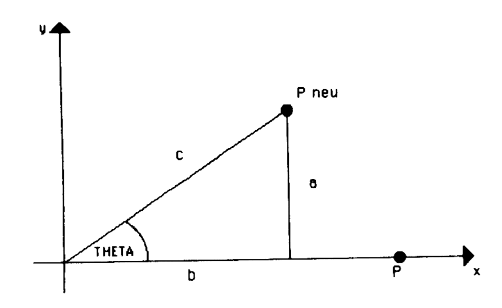
En fait, nous avons déjà réalisé une telle transformation. Il s'agissait de la seconde méthode que nous avons décrite pour tracer un cercle à l'écran. Ce programme ne fait rien d'autre que de tourner autour d'un centre (l'origine du système polaire), pour afficher à distance constante (le rayon) successivement un seul point.

La matrice signifiant une telle rotation peut se trouver en trigonométrie.

La trigonométrie regroupe ce qui à trait à la mesure et au calcul de triangles.
Lorsqu'il s'agit de triangles rectangles, le calcul de toutes les variables se décompose en deux formules et est particulièrement simple.
D'une part, l'addition des angles adjacents à l'hypoténuse donne 90 degrés, d'autre part, vous n'êtes pas sans connaitre le théorème de Pythagore.



Pour appliquer tout ceci, nous pouvons dire que toute figure peut être décomposée en triangles, ce qui est tout particulièrement vrai pour la position d'un point à l'intérieur d'un repère cartésien:



Les fonctions trigonométriques SIN() et COS() forment les rapports entre les cotés d'un triangle rectangle, soit:

$$\mathrm{SIN}(\text{théta}) = \frac{a}{c}$$

$$\mathrm{COS}(\text{théta}) = \frac{b}{c}$$

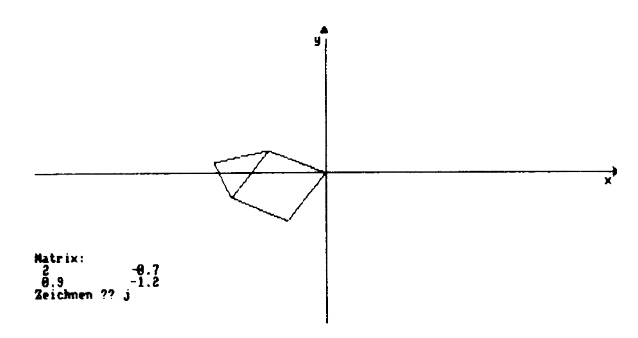
Théta est l'angle de rotation choisi, a représente une partie de l'axe y et b de l'axe x.

Notre matrice devient:

cos(théta)	sin(théta)
-sin(théta)	cos)théta)

Vous êtes impatient d'en faire l'essai; remplacez les lignes suivantes de notre dernier programme:

```
630 xnouveau(point)= COS(a)* x(point)+ (-SIN(c))* y(point)
640 ynouveau(point)= SIN(b)* x(point)+ COS(d)* y(point)
```



utilisation DAO

SHAPES, matrices, transformations de coordonnées s'utilisent non seulement dans des programmes de représentation graphiques d'objets (DAO), mais forment en général le coeur d'impressionnants programmes graphiques sur ordinateur.

Un simple SHAPE, un triangle, un carré ou une forme quelconque sera ici tourné autour d'un point, voir autour de son propre axe:

```
10 REM figure tournante
20 REM ici un carré
30 REM ----------------
40 MODE 2: PAPER 0: PEN 1: INK 0,12: INK 1,0: BORDER 12
45 ORIGIN 320,200
50 x1=-220:x2=80:x3=80:x4=-220
60 y1=-100:y2=-100:y3=100:y4=100
70 sw=5
80 FOR nombre=1 TO 100
90 x=x+sw:y=y+sw
100 x1=x1-sw:x2=x2-sw:x3=x3+sw:x4=x4+sw
110 y1=y1+sw:y2=y2-sw:y3=y3-sw:y4=y4+sw
120 PLOT x1,y1
130 DRAW x2,y2
140 DRAW x3,y3
150 DRAW x4,y4
160 DRAW x1,y1
170 NEXT nombre
```

Les points d'origine ont naturellement une grande influence sur le résultat (lignes 50 et 60), tout comme l'origine du repère cartésien, le nombre de répétitions et le pas.

La petite modification suivante vous donnera une toute autre représentation:

```
50 x1=-220:x2=80:x3=80:x4=-220
60 y1=-100:y2=-100:y3=100:y4=100
```

Mais ce programme est rigide d'utilisation; les données ne sont pas mises dans un tableau et les formules de calculs de transformation ne se trouvent pas du premier coup d'oeil!

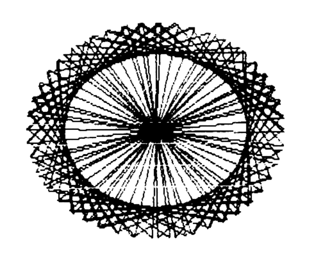
Le programme suivant y remédie:

```
10 REM spirographe
20 REM rotation autour du centre
30 REM -------------------------
40 MODE 2: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,12
50 DEFINT a-z
60 REM origine = centre de l'écran
70 ORIGIN 320,200
80 REM combien de coins?
90 READ nb
100 REM lire les points
110 FOR p=1 TO nb
120 READ xo(p), yo(p)
130 NEXT p
140 REM modification des coordonnées
150 REM tourner combien de fois et de quelle valeur?
160 FOR theta=1 TO 180 STEP 3
170 REM faire calculs de position des points
180 FOR p=1 TO nb
190 x(p)=xo(p)*COS(theta)-yo(p)*SIN(theta)
200 y(p)=xo(p)*SIN(theta)+yo(p)*COS(theta)
210 NEXT p
220 REM dessinner
230 GOSUB 250
240 NEXT theta: CALL &A000
250 PLOT x(1),y(1): DRAW x(2),y(2): DRAW x(3), y(3): DRAW x(1),  y(1):
RETURN
260 REM mettre ici les données de la figure
270 REM d'abord le nombre de coins
280 DATA 3,-100,-100,100,-100,0,100
```

Les données du SHAPE que nous voulons afficher et faire tourner sont comme d'habitude à la fin du programme; afin d'en assurer une bonne exploitation, il ne faut pas omettre de préciser en première position le nombre de coins.
Cete valeur est lue puis attribuée aux variables xo et yo dans les lignes 110 à 130.
L'angle de rotation est défini en ligne 160 et le calcul des équations se fait en 190 et 200.
Nous obtenons ici une rotation dont le résultat est attirant. Et pourquoi ne pas agrandir ou diminuer le SHAPE pendant la rotation?

Là encore, d'autres données transformeront entièrement le dessin:

```
280 DATA 4,-100,-100,100,-100,100,100,-100,100
```



Les innombrables possibilités de rotation de SHAPES sont présentées sans interruption par le programme suivant:

```
10 REM graphique ordinateur2
20 REM rotation autour du centre
30 REM -------------------------
40 MODE 2: PAPER 0: INK 0,12: BORDER 12: PEN 1: INK 1,0
50 DEFINT a-z
60 DIM xo(15),yo(15),x(15),y(15)
70 REM origine = centre de l'écran
80 ORIGIN 320,200
90 REM combien de coins?
100 nb=RND(1)*15
110 REM lire les points
120 FOR p=1 TO nb
130 xo(p)=RND(1)*320-160:yo(p)=RND(1)*200-100
140 NEXT p
150 REM modification des coordonnées
160 REM tourner combien de fois et de quelle valeur?
170 thetamax=RND(1)*360: sw=RND(1)*10
180 FOR theta=1 TO thetamax STEP sw
190 REM faire calculs de position des points
200 FOR p=1 TO nb
210 x(p)=xo(p)*COS(theta)-yo(p)*SIN(theta)
220 y(p)=xo(p)*SIN(theta)+yo(p)*COS(theta)
230 NEXT p
240 REM échelle=échelle*1.1
250 REM dessiner
260 GOSUB 290
270 NEXT theta
280 FOR i=1 TO 5000: NEXT: CLS: GOTO 100
```

```
290 PLOT x(1),y(1): FOR p=1 TO nb: DRAW x(p),y(p): NEXT p: DRAW  x(1),
y(1): RETURN
```

fonctions

Une autre utilisation importante du graphisme en deux dimensions sur ordinateur est la représentation graphique, point par point, de fonctions.
En d'autres termes, la relation entre 2 valeurs est schématisée par l'intermédiaire d'une formule mathématique.

L'utilisation de fonctions remonte au 18e siècle.
Mais la définition alors donnée ne devait pas suffire a l'évolution des mathématiques. On parlait alors de 'valeur variable dépendant d'une autre valeur également variable'.
Ce qui caractérise les fonctions,ce ne sont pas les différentes valeurs numériques, mais bien plus leur relation entre elles.
Ceci explique que non seulement des chiffres peuvent être mis en fonction, mais également des objets par exemple.

Naturellement, un mathématicien ne parlera jamais d'un rang, mais d'une quantité. Il s'expliquera de la façon suivante:

'une fonction classe chaque élément d'une quantité selon un élément défini d'une autre quantité.'

Comme exemple, nous pourrions prendre d'une part un nombre de voitures et d'autre part un même nombre de conducteurs. En en faisant la relation il serait possible d'attribuer un conducteur à chaque automobile!
Cette relation n'est pas encore une fonction; il n'est pas possible de définir le conducteur allant avec une voiture donnée.

Il serait déjà plus simple de faire cette relation par rapport aux titulaires des cartes grises, une seule personne correspondant bien à une voiture précise.

Appelons x les automobiles et y leurs propriétaires. La fonction peut

maintenant être formulée:

La représentation graphique d'une fonction est la schématisation de deux quantités, c'est à dire d'une quantité donnée de valeurs ordonnées (x,y), dans la mesure ou pour chaque x il y ait une valeur y correspondante.

Soit: $y = f(x)$

x est l'élément de la première identité, soit des valeurs connues alors que y représente la valeur recherchée.
On parle aussi de plages de valeurs et définitions.

représentation des fonctions

Tout étudiant apprend de nos jours au moins trois méthodes différentes de représentation graphique de fonctions.

Nous pouvons tout de suite oublier le type de représentation par cercle et ovale sur lesquels les valeurs sont mises en relation par de petites flèches.
De chaque valeur donnée part donc une seule flèche; en contre-partie, plusieurs flèches peuvent désigner l'un des éléments recherchés.

Une table de valeurs est facilement réalisable en partant d'un tel graphique:
chaque élément donné est inscrit sur une colonne, les valeurs flèchées sont ensuite mises en correspondance.

Nous pouvons affecter à chaque groupe de 2 valeurs un point 'p' sur un plan et obtenons ainsi une représentation graphique.

En général, ce sont les repères cartésiens qui sont utilisés, les valeurs 'x' recherchées peuvent être lues sur l'axe hotizontal et les valeurs données 'y' sur l'axe vertical.

Selon la définition et la fonction elle-même, nous obtenons une série de points, courbes ou une belle ligne la représentant.

tracé de fonctions

Il ne nous reste plus qu'à transposer tout ceci dans un programme en tenant compte des caractéristiques de l'ordinateur.

Le premier pas est la réalisation d'une table de valeurs:

```
10 PAPER 0: PEN 1: BORDER 12: INK 0,12: INK 1,0
200 FOR x=1 TO 10
210 y=x*x
220 PRINT x;y
230 NEXT x
```

La ligne 200 définit la résolution du graphique, son pas; dans ce cas de 1 à 10. La fonction est inscrite et calculée en ligne 210. La ligne 220 enfin, elle permet d'inscrire les unes sous les autres les valeurs sous forme d'un tableau.

Une représentation simplifiée est déjà possible; remplacez les lignes suivantes:

```
220 PRINT TAB(y)"*"
```

et ajoutez

```
20 MODE 2
```

Lancez ce programme et vous reconnaitrez sans difficulté le tracé d'une parabole connue.

Vous auriez d'ailleurs procédé de la même façon sur papier.
Vous auriez tracé un repère cartésien sur lequel les valeurs x iraient en croissant de gauche à droite et les valeurs y de bas en haut.
Vous auriez ensuite inscrit les différents points sur les axes puis

mis en rapport les points recherchés puis les auriez reliés d'un trait.

Nous l'avons remarqué, notre ordinateur AMSTRAD CPC à un comportement humain; il se dirige sur l'écran comme nous sur une feuille; il permet aussi un libre choix du point d'origine des coordonnées.

Dans les précédentes pages, nous avons déjà tracé les axes, inscrit leurs designations x et y qui ne nous sont pas inconnues; il devrait suffire de complèter ce programme des précédentes routines et de modifier la ligne 220 par une instruction PLOT(x,y):

```
10 PAPER 0:PEN 1:BORDER 12:INK 0,12:INK
1,0
20 MODE 2
100 ORIGIN 320,200:CLG:TAG
110 PLOT -320,0:DRAW 320,0
120 MOVER -5,6:PRINT CHR$(246);
130 MOVER -17,-10:PRINT"x";
140 PLOT 0,-200:DRAW 0,200
150 MOVER -4,-2:PRINT CHR$(244);
160 MOVER -17,-10:PRINT"y";
170 TAGOFF
200 FOR x=-10 TO 10
210 y=x*x
220 PLOT x,y
230 NEXT x
```

Le tracé se fait comme nous l'attendions sur l'écran, seul sa dimension serait à parfaire.

Plus un problème pour nous; nous avons vu qu'une mise à l'échelle s'obtient par simple multiplication par un facteur constant.

Agrandissons donc notre parabole en prenant un facteur 20.

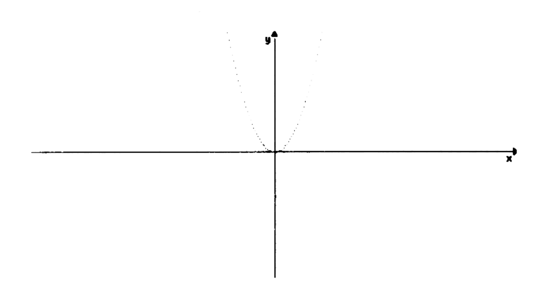
```
180 mf=20
220 PLOT x*mf,y*mf
200 FOR x=-3 TO 3 STEP 0.01
```

Le dessin laisse maintenant supposer que tout est en ordre. Mais il faut bien connaitre la fonction pour la reconnaitre avec si peu de

points.

Il nous faut plus de points, pour ne pas dire le plus possible.

Complétons la ligne 200 par STEP .1 et tout s'améliore:



Pour obtenir une ligne continue, il nous reste deux solutions: soit nous faisons le calcul des positions restantes à compléter, soit nous faisons comme sur notre feuille de papier et faisons tracer par l'ordinateur les droites entre les points à relier.

Cette dernière solution est aussi rapide qu'imprécise puisque notre ordinateur ne fait pas d'approximation dans les courbes du tracé comme nous le ferions à la main, mais il tracera des droites.

Le premier programme est précis, mais quelle lenteur vu le nombre de calculs à effectuer; plus le pas est petit, plus il y aura de calculs.

Vous pouvez vous en assurer en changeant les lignes 200 et 210:

```
200 FOR x=-10 TO 10 STEP 0.01
210 y=SIN(x)
```

Qu'en serait-il d'une combinaison des deux solutions? Et pendant que nous y sommes, entrons dans ce programme les différentes valeurs de

contrôle désirées, nous permettant d'agrandir le graphique selon nos souhaits si les détails ne sont pas suffisants.

Entrez le listing suivant et essayez ce programme.

N'oubliez pas de remplacer en ligne 10000 la fonction par d'autres, vous obtiendrez autre chose que la parabole précédente.

```
10 PAPER 0: PEN 1: BORDER 12: INK 0,12: INK 1,0
20 MODE 2
100 REM axes
110 ORIGIN 320,200: CLG: TAVG
120 PLOT -320,0: DRAW 320,0
130 MOVER -5,6:PRINT CHR$(246);
140 MOVER -17,-10: PRINT"x";
150 PLOT 0,-200: DRAW 0,200
160 MOVER -4,-2:PRINT CHR$(244);
170 MOVER -17,-10: PRINT"y";
180 TAGOFF
190 REM entrée des données
200 INPUT"de x= ";x: xmin=x: GOSUB  10000:  ymin=x:  INPUT"jusqu'à  x=
";x: xmax=x: GOSUB 10000: ymax=x: INPUT"pas (précision) ";précision
210 INPUT"facteur d'agrandissement ";mf
300 REM tracé de la fonction
310 x=xmin: GOSUB 10000: PLOT x*mf,y*mf: REM premier point
320 FOR x=xmin TO xmax STEP précision
330 GOSUB 10000
340 DRAW x*mf,y*mf
350 NEXT x
360 GOSUB 410: REM mise à l'échelle
370 END
400 REM mise à l'échelle de l'axe des x
410 FOR x=xmin TO xmax STEP 10
420 GOSUB 10000
430 PLOT x*mf,5: DRAW x*mf,-5
440 MOVE 0,-10: PRINT INT(x);
450 NEXT x
460 RETURN
9990 REM  indiquer  en  ligne  10000  la  formule  de  la  fonction  à
```

représenter.

```
10000 y=SIN(x)
10010 RETURN
```

En principe, tout fonctionne bien. Il ne manque plus qu'une inscription sur les axes pour rendre cette représentation graphique d'une fonction encore plus explicite.

Là, il y a encore plusieurs techniques; nous vous présentons ici l'une d'elles.

Définissons en premier lieu le nombre de graduations de chaque axe.

Dans notre cas, ce seront dix, soit une tous les 64 points sur l'axe x. Pour l'axe des y, la distance sera de 40 points.

Ces considérations terminées, écrivons une boucle qui se chargera de la graduation des axes.

La valeur de chacune d'elles sera le quotient du nombre total de points par le facteur d'échelle.

Le calcul terminé, l'inscription peut être faite avec TAG, après positionnement du curseur graphique.
N'oublions pas d'activer le mode 'branchement conditionnel' pour éviter d'effacer certaines parties du graphique.

Un dernier problème peut survenir lors du calcul des différentes valeurs, en cas de division par 0.
Cette erreur devrait arrêter immédiatement le programme. Pour éviter cela, nous additionnerons à chaque valeur servant au calcul une constante (0.00001 par exemple). Nous pouvons aussi effectuer un contrôle faisant continuer le programme par le calcul suivant si cette erreur survient.

```
5 REM traceur
10 ON ERROR GOTO 20000: REM dans le doute!
20 PAPER 0: PEN 1: BORDER 12: INK 0,12: INK 1,0
30 MODE 2
50 REM entrée des données
60 INPUT"de x= ";x: xmin=x: GOSUB 10000: ymin=x: INPUT"jusqu'à x= ";x:
xmax=x: GOSUB 10000: ymax=x: INPUT"pas (précision) ";précision
70 INPUT"facteur d'agrandissement ";mf
90 MODE 2
100 REM axes
110 ORIGIN 320,200: CLG: TAG
120 PLOT -320,0: DRAW 320,0
130 MOVER -5,6:PRINT CHR$(246);
140 MOVER -17,-10: PRINT"x";
150 PLOT 0,-200: DRAW 0,200
160 MOVER -4,-2:PRINT CHR$(244);
170 MOVER -17,-10: PRINT"y";
180 TAGOFF
300 REM tracé de la fonction
310 x=xmin: GOSUB 10000: PLOT x*mf,y*mf: REM premier point
320 FOR x=xmin TO xmax STEP précision
330 GOSUB 10000
340 DRAW x*mf,y*mf
350 NEXT x
360 GOSUB 400: REM mise à l'échelle
370 IF INKEY$="" THEN 370 ELSE 20
390 REM échelle de l'axe des x
400 PRINT CHR$(23); CHR$(1);: TAG: REM écrire en superposition
410 FOR xa=-320 TO 320 STEP 64
420 x=xa/mf: GOSUB 10000
430 PLOT xa,5: DRAW xa,-5
```

```
440 MOVER -19,-10: PRINT x;
450 NEXT xa
490 REM échelle de l'axe des y
500 FOR ya=40 TO 200 STEP 40
510 y=ya/mf
520 PLOT -5,ya: DRAW 5,ya
530 MOVER -1,6: PRINT y;
540 NEXT ya
550 REM axe y
560 FOR ya=-40 TO -200 STEP -40
570 y=ya/mf
580 PLOT -5,ya: DRAW 5,ya
590 MOVER -1,6: PRINT y;
600 NEXT ya
610 TAGOFF
620 PRINT CHR$(23); CHR$(0);: REM fin d'écriture en superposition
630 RETURN
9990 REM indiquer en ligne 10000 la formule de la fonction à
représenter.
10000 y=SIN(x)/x
10010 RETURN
20000 RESUME NEXT: REM continuer
```

Chapitre 5

GRAPHISME EN 3D

la troisième dimension

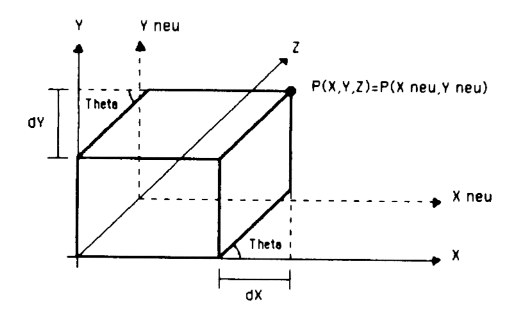
Toute une série de techniques se sont développées au cours des dernières années dans le domaine de la représentation graphique tridimensionnelle. Certaines de ces techniques sont adaptées à l'informatique.

En fait, il s'agit toujours d'un effet, d'une illusion de trois dimensions; comment, sur une surface plane obtenir un endroit plus profond qu'un autre?
La majorité des dessins 3D sont donc des vues en perspectives.

vues en perspective

Ce type d'image en 3D comme nous en voyons quotidiennement à la télévision et sur des photographies ne demande aucun artifice technique, seulement un peu de pratique des mathématiques.

Pour obtenir cet 'effet d'optique', tous les points par exemple d'une maison à représenter, seront projetés sur un seul plan.



Les points du troisième axe 'z' sont représentés le long de l'axe des x.
Le schéma précédent montre bien que pour afficher le point arrière droit du dé, une modification de coordonnées était nécessaire, qui avait la valeur correspondante à la distance de P(x,y,z) à l'axe x, soit exactement z.

Dans le nouveau système polaire, xnouveau et ynouveau, le point P(x,y,z) sera précisé seulement à l'aide de deux coordonnées.

En regardant ce graphisme de plus près, on s'aperçoit que nos connaissances devraient nous permettre de créer de telles images en perspective.
Dans le chapitre précédent, nous avons vu une formule pouvant être utilisée dans ce cas.

La position de l'axe z sera déterminée dès le début, elle sera dans un rapport constant par rapport à l'axe x; l'angle que forment les deux axes est donc connu.

La longueur de ce nouvel axe est aussi connue, ce qui nous permet d'indiquer les coordonnées du nouveau point P(x,y) découlant de P(x,y,z):

x = X2 + z * COS(theta)

y = Y2 + z * SIN(theta)

tracer en trois dimensions

L'exactitude de ce que nous venons de dire peut être vérifiée très rapidement à l'aide du programme ci-dessous.
Une variable a été mise en fonction d'une autre, ici nous voulons calculer z en fonction de x et y.

Nous allons donc construire une seconde boucle autour de la première.

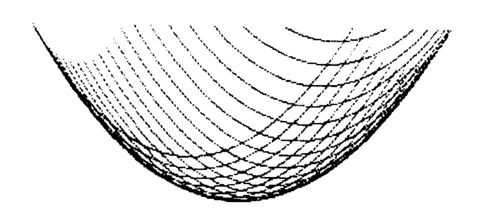
D'autre part, pour chacun des axes nous choisissons une échelle différente, et pouvons ainsi agrandir et diminuer le graphique dans tous les sens.

```
10 PAPER 0:PEN 1:BORDER 12:INK 0,12:INK
1,0
20 MODE 2
30 ORIGIN 320,200
60 MFX=30:MFY=30:MFZ=5
70 FOR Y=-6 TO 6 STEP 0.5
80 FOR X=-6 TO 6 STEP 0.05
90 GOSUB 10000
100 SX=X*MFX+Y*MFY*COS(45)
110 SY=Z*MFZ+Y*MFY*SIN(45)
120 PLOT SX,SY
130 NEXT X
140 NEXT Y
150 END
10000 Z=X*X+Y*Y
10010 RETURN
```

Vous remarquerez sans difficulté la formule de transformation permettant le calcul des coordonnées en ligne 100 et 110

En ligne 10000, nous avons placé la fonction usuelle représentant notre parabole, cette fois-ci en 3 dimensions.

Le graphique obtenu par ce programme devrait être similaire à une éprouvette.



L'écartement des lignes est défini par le pas en ligne 70 et 80.

La taille du graphique peut être modifiée en ligne 60

Vous devriez aussi examiner l'efficacité de ce programme.

Car l'angle, dont les sinus et cosinus sont constamment recalculés, définit l'axe z et reste constant pendant le déroulement du programme.

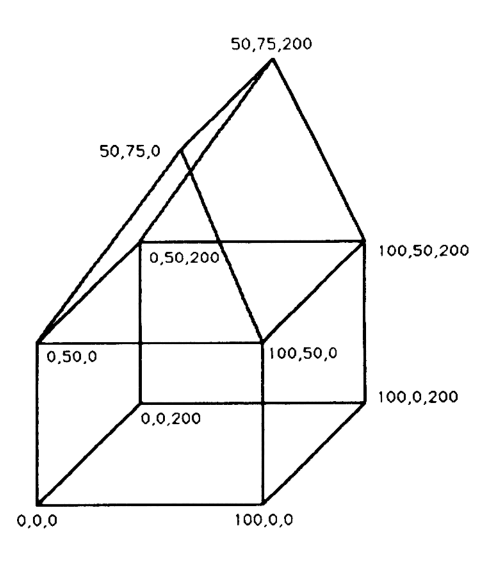
Les calculs trigonométriques auront toujours le même résultat. Nous pouvons donc entrer une constante en ligne 100 et 110.

```
10 REM tracer en 3D
20 PAPER 0: PEN 1: BORDER 12: INK 0,12: INK 1,0
30 MODE 2
40 ORIGIN 320,200
50 REM échelle pour tous les axes.
60 INPUT"facteur x ";mfx: INPUT"facteur y";mfy: INPUT"facteur z ";mfz
70 REM angle de l'axe z
80 w1=COS(45): w2=SIN(45)
90 REM tracer fonction
100 FOR y=-4 TO 4 STEP 0.25
110 FOR x=-4 TO 4 STEP 0.01
120 GOSUB 10000
130 REM calcul coordonnées
140 sx=x*mfx+y*mfy*w1
150 sy=z*mfz+y*mfy*w2
160 PLOT sx,sy
170 NEXT x
180 NEXT y
190 END
9990 REM fonction en ligne 10000
10000 z=x*x+y*y
10010 RETURN
```

Le prochain pas à franchir est naturellement de porter à l'écran des objets réels.
Nos formules le permettent, mais l'objet en question devrait tout d'abord être mis en valeurs numériques qui seraient la base de toute opération.

Cette opération pourrait se résumer comme suit: l'objet serait mesuré sous tous ses aspects, l'un des coins serait pris comme point 0 et la distance des autres points mesurée par rapport a ce 0.

La maison que nous avons souvent utilisée dans nos exemples serait décrite de la façon suivante:



Ces coordonnées doivent être transmises à l'ordinateur pour en faire le traitement. Il faut être rigoureux et bien définir la relation entre les points.
Les points seuls ne font en effet pas le dessin, ce sont les traits qui relient ces points.

Il serait plus judicieux de donner pour chaque ligne ses coordonnées de début et de fin; toute suite logique dans le tracé étant ainsi superflue.

Puis le dessin sera plus facile à afficher si chaque ligne est agrandie, diminuée, déplacée ou effacée individuellement.

Notre maison se compose de 17 lignes; la table de valeurs se présente comme suit:

```
1010 DATA 0,0,0,10,0,0
1020 DATA 10,0,0,10,5,0
1030 DATA 10,5,0,0,5,0
1040 DATA 0,5,0,0,0,0
1050 DATA 10,0,20,10,5,20
1060 DATA 10,5,20,0,5,20
1070 DATA 0,5,20,0,0,20
1080 DATA 0,0,20,10,0,20
1090 DATA 10,0,0,10,0,20
1100 DATA 0,0,0,0,0,20
1110 DATA 0,5,0,0,5,20
1120 DATA 10,5,0,10,5,20
1130 DATA 10,5,0,5,7,0
1140 DATA 0,5,0,5,7,0
1150 DATA 5,7,0,5,7,20
1160 DATA 5,7,20,0,5,20
1170 DATA 5,7,20;10,5,20
```

Si l'on indique avant tout le nombre de lignes, toutes les données peuvent être mises dans un tableau de variables qui sera lu sous la forme de xde, yde, zde et xvers, yvers et zvers.

```
100 FOR L=1 TO 17
110 READ XV(L),YV(L),ZV(L),XN(L),YN(L),Z
N(L)
120 NEXT L
```

L'objet défini se trouve ainsi en mémoire, en trois dimensions!
Avant de faire l'affichage, un certain nombre de calculs seront nécéssaires pour obtenir les coordonnées x et y.

Les équations connues peuvent être employées:

```
200 FOR L=1 TO 17
210 SXV(L)=XV(L)*ZV(L)*COS(45):SXN(L)=XN
(L)+ZN(L)*COS(45)
220 SYV(L)=YV(L)+ZV(L)*SIN(45):SYN(L)=YN
(L)+ZN(L)*SIN(45)
230 NEXT L
```

Après ceci, plus rien ne nous empèchera de tracer notre dessin ligne par ligne:

```
300 ORIGIN 320,200:FOR L=1 TO 17
310 PLOT SXV(L),SYV(L):DRAW SXN(L),SYN(L
)
```

perspective

Naturellement, les coordonnées des points une fois en mémoire peuvent être manipulés dans toutes les règles de l'art.

En principe, cette maison n'est rien d'autre qu'un SHAPE, sous un aspect légèrement différent de ce que nous avons vu au chapitre 4.
La seule différence est que chaque point,hormis le premier, était le dernier point d'une ligne alors qu'ici, c'est le premier.

Mais l'utilisation des SHAPES reste identique, ce que montre encore une fois le programme suivant. Celui-ci est un véritable petit programme de DAO, surtout en ce qui concerne les possibilités de mise à l'échelle et de déformation.

```
10 REM DAO
20 MODE 2: ORIGIN 320,200
30 mfx=5: mfy=5: mfz=5: w=45
40 ORIGIN 320,200
50 REM combien de lignes?
60 READ nb
70 REM tableau des coordonnées
80 DIM sxv(nb), sxn(nb), syn(nb), syv(nb), xv(nb), yv(nb), zv(nb),
xn(nb), yn(nb), zn(nb)
90 REM lecture des points
100 FOR l=1 TO nb
110 READ xv(l), yv(l), zv(l), xn(l), yn(l), zn(l)
120 NEXT l
190 REM calcul en coordonnées écran
200 FOR l=1 TO nb
210 sxv(l)=e+xv(l)*mfx+zv(l)*COS(w)*mfz:
sxn(l)=xn(l)*mfx+zn(l)*COS(w)*mfz
220 syv(l)=yv(l)*mfy+zv(l)*SIN(w)*mfz:
syn(l)=yn(l)*mfy+zn(l)*SIN(w)*mfz
230 NEXT l: e=0
240 GOTO 400
290 REM tracer
300 ORIGIN 320,200: FOR l=1 TO nb
310 PLOT sxv(l), syv(l): DRAW sxn(l), syn(l)
320 NEXT l
330 IF INKEY$ ="" THEN 330 ELSE 400
390 REM menu
400 CLS: PRINT"vous souhaitez:": PRINT: PRINT
410 PRINT"1 - agrandir/ diminuer
420 PRINT"2 - déformation
430 PRINT"3 - nouvel angle de vue
```

```
440 PRINT"4 - dessin d'origine
450 PRINT"5 - affichage du dessin
460 PRINT"6 - terminer
470 a$=INKEY$: IF a$="" THEN 470
480 a=VAL(a$)
490 IF (a<0 OR a>6) THEN 400
500 ON a GOTO 600,700,800,820,900,910
590 REM échelle
600 CLS: INPUT "pour un agrandissement, entrer un chiffre>5 , pour
une diminution, une valeur comprise entre 0 et5";mfg
610 mfx=mfg: mfy=mfg: mfz=mfg
620 GOTO 200
690 REM déformation
700 CLS: INPUT"dans quelle direction: x=1 y=2 z=3 ";r: INPUT"entrer
pour une extension un chiffre>5 et pour une contraction une valeur
entre 0 et5";mf
710 IF (r<0 OR r>3) THEN 700
720 ON r GOTO 730, 740, 750
730 FOR l=1 TO nb: xv(l)= xv(l)*mf:xn(l)=xn(l)*mf: NEXT l: GOTO 200
740 FOR l=1 TO nb: yv(l)= yv(l)*mf:yn(l)=yn(l)*mf: NEXT l: GOTO 200
750 FOR l=1 TO nb: zv(l)= zv(l)*mf:zn(l)=zn(l)*mf: NEXT l: GOTO 200
790 REM axe z
800 CLS: INPUT" degrés entre les axes x et z ";w
810 GOTO 200
820 CLS: RUN
890 REM affichage dessin
900 MODE 2: GOTO 300
910 CLS: END
980 REM données de l'objet selon: point de départ de ligne (x,y,z), et
de fin (x,y,z)
```

Chapitre 6

SON

le son fait la musique

Il ne s'agit pas seulement d'une façon de parler; vous allez le voir, justement dans le domaine de la création musicale par le biais de l'électronique, cette formule prend toute sa signification.

Chacun sait qu'un son est une vibration. Celle-ci, portée par les airs arrive à nos oreilles et met en vibration plus ou moins rapide notre tympan.
Cette vibration, transformée en impulsion électrique, donne à notre cerveau l'impression d'un son. Plus les vibrations de notre tympan sont rapides, plus le son sera perçu comme aigu.

Mais comment expliquer qu'une corde qui vibre 440 fois par seconde dans un piano droit rende un son différent de celui d'une corde de même longueur dans un piano à queue?

Simple répondront certains, un instrument à une attaque douce et l'autre une attaque dure.

Mais quelle en est l'influence sur la sonorité?

Pour répondre à cette question, la construction du son doit être connue.
On différencie principalement les sons réguliers, périodiques des bruits irréguliers.

Nous avons bien dit d'une part sons et d'autre part bruits, cela est assez explicite; en exemple nous pouvons citer les sons provenant d'une flûte à bec, et les bruits de souffle de l'air.

Aucun son produit par un instrument n'est absolument pur, il se décompose toujours en plusieurs sons, en fréquence de base et harmonique.

Ces relations ont été établies au début du 19e siècle par le

mathématicien français Joseph de Fourier dans son oeuvre 'théorie analytique de la chaleur'.
Il s'est attaché à représenter des fonctions périodiques. Ses théories furent la base de nombreuses utilisations mathématiques, physiques et techniques.

Il développa ainsi le procédé de l'analyse harmonique des sons, qui décompose une vibration en une série de sinusoïdales pures et une part de constantes.
Il constata que les fréquences harmoniques toujours présentes avec la fréquence de base, sont un multiple de celles-ci.

En tenant compte de ces considérations, il est possible de suivre la démarche inverse, la synthèse harmonique.
Chaque vibration est additionnée, ce qui donne une résultante.
De cette sorte on peut construire d'autres formes de sons, carré, en dent de scie...

Outre la fréquence de base qui définit la hauteur du son, les harmoniques déterminent la forme de l'onde, la sonorité de la note.

le synthétiseur

Ce sont ces bases qui ont servies à Bob Moog lors de la présentation du synthétiseur Moog en 1964.
Il avait construit un monstre technique permettant de modifier les composantes d'un son, la fréquence, l'harmonique et le volume.

Environ 20 ans après, un seul circuit intégré, à commande digitale, disposait de caractéristiques similaires et même parfois supérieures à celles du synthétiseur Moog.

Le CPC 464 utilise un circuit intégré AY-3-8912 de Général Instruments. Il permet de jouer sur 3 voix, ce qui correspond à un joueur de piano n'ayant que trois doigts, mais autorise tout de même

certaines prouesses.

En plus, le branchement des canaux est réalisé pour rendre un effet stéréophonique si vous êtes connecté à une chaine stéréo. Les sons envoyés sur le canal A sortirons sur le canal de gauche, ceux du canal C sur le canal droit et ceux du canal B au milieu, soit sur les deux enceintes.

Le volume sonore peut se régler en 15 positions différentes, tout comme la durée , la tonalité, la fréquence, et ainsi de suite!

le son - SOUND

Les réglages de base les plus importants se font par l'instruction SOUND

Ce sont:

-choix du générateur de sons
-choix de la note
-indication de la durée
-indication du volume sonore

Vous pouvez d'autre part affecter à chaque son une

-enveloppe de volume
-enveloppe de tonalité
-bruit

Le choix du canal

C'est le premier paramètre de la commande SOUND.
SOUND 1 = canal A
SOUND 2 = canal B
SOUND 3 = canal C

Il sera souvent nécessaire d'utiliser simultanément plusieurs canaux; pour ceci, il suffit d'additionner les valeurs des canaux correspondants.
SOUND 5 permettra de sortir les notes simultanément sur les canaux B et C, SOUND 3, sur A et B.

Cela explique pourquoi il n'est pas possible de se faire suivre les instructions SOUND 1..., SOUND 2..., SOUND 3...

Le premier paramètre de l'instruction SOUND permet en outre de synchroniser les canaux, la technique de rendez-vous, tout comme de

marquer des points d'arrêt relevés par RELEASE.

SOUND	conséquence
1	uniquement canal A
2	uniquement canal B
4	uniquement canal C
8	synchro avec canal A
16	synchro avec canal B
32	synchro avec canal C
64	arrêt
128	bruit

la fréquence

Elle sera donnée selon une période. Si vous souhaitez préciser une note, votre manuel d'emploi vous en donne les équivalences; ce sera le deuxième paramètre de l'instruction SOUND. La fréquence est calculée environ avec 125000/période.

Si vous êtes impatient d'utiliser le générateur de sons, ou de vérifier la bande passante de votre magnétophone, essayez donc ce programme:

```
10 MODE 1
20 PERIODE=10
30 E$=INKEY$:IF E$<>"" THEN PERIODE=PERI
ODE+10
40 LOCATE 1,1:PRINT"PERIODE: ";PERIODE;"
FREQUENZ CA. ";INT(125000/PERIODE)
50 SOUND 1,PERIODE
60 GOTO 30
```

La durée du son

Elle sera donnée par le troisième paramètre dans SOUND.
Il n'est malheureusement pas possible d'indiquer la durée d'une note en 1/4 ou 1/8, ces valeurs étant affectées par le temps.
Le concepteur du CPC à prévu une entrée en 1/100 de secondes.

Si vous entrez un morceau de musique, prenez d'abord les valeurs 25, 50 et 75. Après une écoute critique vous pourrez corriger selon votre oreille.

Grace à ces seules connaissances, une serie d'expériences vous attend sur le CPC.

Entrez un morceau de musique, programmez en un, ou encore construisez-vous un petit orgue.

Après avoir entré le programme suivant, modifiez en les différents paramètres de l'instruction SOUND et jugez en par vous même.

Le son sera meilleur si vous faites jouer les trois canaux au lieu d'un seul.

```
10 REM Londonderry-Air
20 READ a,b,c
30 IF b=-1 THEN RESTORE:GOTO 20
40 SOUND 1,a,b,c
50 GOTO 20
60 DATA 338,50,5,319,50,5,284,50,5,253,1
00,5,284,50,5,253,50,5,190,50,5
70 DATA 213,50,5,253,50,5,284,50,5,319,5
0,5,379,75,5,379,50,5,319,50,5
80 DATA 253,50,5,239,50,5,213,100,5,190,
50,5,213,50,5,253,50,5,319,50,5
90 DATA 253,50,5,284,150,5,284,50,5,338,
50,5,319,50,5,284,50,5,253,100,5
100 DATA 284,50,5,253,50,5,190,50,5,213,
50,5,253,50,5,284,50,5,319,50,5
110 DATA 379,75,5,379,50,5,338,50,5,319,
50,5,284,50,5,253,100,5,239,50,5
120 DATA 253,50,5,284,50,5,319,50,5,284,
50,5,319,150,5,319,50,5,213,50,5
130 DATA 190,50,5,169,50,5,159,100,5,169
,50,5,169,50,5,190,50,5,213,50,5
140 DATA 190,50,5,213,50,5,253,50,5,319,
75,5,319,50,5,213,50,5,190,50,5
150 DATA 169,50,5,159,100,5,169,50,5,169
,50,5,190,50,5,213,50,5,253,50,5
160 DATA 284,150,5,284,50,5,213,50,5,213
,50,5,213,50,5,127,100,7,142,50,7
170 DATA 142,50,7,159,50,7,190,50,7,159,
50,7,213,50,7,253,50,7,319,75,7
180 DATA 319,50,7,338,50,7,319,50,7,284,
50,7,253,50,7,190,50,7,213,50,7
190 DATA 253,50,7,284,50,7,319,50,7,379,
50,7,338,50,7,319,175,7
200 DATA 0,-1,0
```

Un orgue

Votre orgue aura besoin d'un clavier. Celui de l'ordinateur s'y prête très bien.
Vous pouvez choisir et prendre dans la deuxième rangée les touches S à L auxquelles vous affecterez une note d'une octave.

Le manuel d'emploi nous donne les valeurs suivantes:

s	C	478
d	D	451
f	E	379
g	F	358
h	G	319
j	A	284
k	B(H)	253
l	C	239

Le programme deviendra:

```
100 e$=INKEY$
110 IF e$="s" THEN periode=478:GOTO 220
120 IF e$="d" THEN periode=426:GOTO 220
130 IF e$="f" THEN periode=379:GOTO 220
140 IF e$="g" THEN periode=358:GOTO 220
150 IF e$="h" THEN periode=319:GOTO 220
160 IF e$="j" THEN periode=284:GOTO 220
170 IF e$="k" THEN periode=253:GOTO 220
180 IF e$="l" THEN periode=239:GOTO 220
210 GOTO 100
220 SOUND 1,periode:GOTO 100
```

Vous pouvez jouer sur un seul octave, une légère modification vous permet d'en changer.

Nous utiliserons pour ceci la touche - ainsi que celle à coté, avec la flèche, pour monter ou descendre d'un octave.

A chaque octave, la période est doublée ou divisée par deux:

```
80 o=1
190 IF e$="-" THEN o=o*2
200 IF e$="^" THEN o=o/2
220 SOUND 1,periode*o:GOTO 100
```

accord automatique

Si vous souhaitez faire évoluer votre 'orgue', grâce au BASIC, on peut même envisager un accord automatique.

En utilisant la routine d'interruption, l'ordinateur peut être forcé de traiter une sous-routine tous les 1/50 secondes.
Cette sous-routine peut bien entendu comprendre l'instruction SOUND pour un autre canal qui se chargera de l'accompagnement; un autre canal restera libre pour notre jeu au clavier.

Complétez le programme:

```
90 EVERY 30 GOSUB 230
230 SOUND 4,426:SOUND 4,284:RETURN
```

Chaque canal dispose d'une petite mémoire mettant en attente jusqu'à 5 notes restantes à jouer; notre accompagnement peut donc jouer une petite mélodie.
Remplir cette file d'attente est rapide, le détournement de l'interruption sur notre sous-routine ne se fera pas sentir.
Ici aussi, essayez différentes solutions, il vous reste un canal de libre...

musique à voix multiples

Le principe est le même.
arrêtez le programme, effacez la ligne 90 et modifiez les lignes 220 et 230 comme suit:

```
220 SOUND 1,periode*o
230 SOUND 4,periode/o:GOTO 100
```

Chaque son joué sortira maintenant à deux niveaux différents.
Mais nous ne pouvons garantir la simultanéïté du son sur les canaux.
Cet exemple ne permet pas de s'en rendre compte, chaque son étant envoyé sur un canal, mais en plus alternativement, l'un après l'autre.
Malgré tout, assurez vous de la synchronisation et ne laissez pas le hasard s'en charger!

synchronisation

Nous voici de retour sur ce sujet qui n'est pas dénué d'intérêt.

Vous savez déjà qu'en fonction de la synchronisation souhaitée, vous devez additionner 8,16 ou 32 au code du canal désiré.

Ce qui n'est pas clair dans le manuel d'emploi du CPC est que

l'instruction de synchronisation doit être donnée aux deux canaux.

Il n'est pas suffisant d'entrer SOUND 4+8,248 en espèrant que la note sortira en même temps que sur le canal A.

L'instruction devrait être:

```
SOUND 1+32,période
SOUND 4+8,248
```

Le canal C doit donc savoir qu'il est synchronisé avec le canal A et vice-versa.

Il est difficile de modifier des airs programmés comme 'Londonderry' pour changer de canal ou utiliser un canal complémentaire.
Cette technique sera adaptée quand la file d'attente de chaque canal peut être remplie en un seul coup.

C'est le moyen le plus effectif avec lequel l'on obtient les meilleurs effets musicaux pour le minimum de programmation. Mais ces suites de notes qui sont envoyées par 'paquets' à chaque canal doivent comporter des points de synchronisation.

Faites un essai en envoyant sur chaque canal 5 notes, toutes de durée différente.
Sans synchronisation, chaque canal jouera les notes les unes après les autres. Avec des points de synchronisation, tout sera en accord.

Le déroulement dans le temps est encore schématisé ci-dessous:

CANAL A	CANAL B	CANAL C
--B--	-CA--	-B---

En jouant la première note, chaque canal commence indépendamment des autres. Le début de la deuxième note est synchronisé entre les canaux B et C.

C peut continuer avec la troisième note alors que A et B doivent s'attendre.

Les séquences suivantes se remarqueront dans le programme:

```
SOUND 1,...      SOUND 2,...      SOUND 4,...
SOUND 1,...      SOUND 34,...     SOUND 20,...
SOUND 17,...     SOUND 10,...     SOUND 4,...
SOUND 1,...      SOUND 2,...      SOUND 4,...
```

le volume sonore

Pour en terminer avec les instructions SOUND, passons au quatrième paramètre, le volume sonore qui peut être règlé de 0 -rien- à 7 -fort-.
Sauf si vous avez précisé avec les paramètres suivants une enveloppe de volume, le volume maximum peut être indiqué par 15.

l'enveloppe de volume

Elle permet de changer la sonorité de telle sorte que l'on peut lui attribuer un instrument.

Imaginez vous souffler dans une trompette.Le son n'atteint pas son plein volume dès l'ouverture du piston, mais est fonction de la pression d'air; il montera, se stabilisera puis baissera soit rapidement si vous refermez le piston, soit lentement, en parallèle à votre expiration.

Vu dans le détail, il y a 4 stades dans l'évolution d'un son.

Le terme anglo-américain est 'attack, decay, sustain, release'; nous français disons 'attaque, chute, maintient, relache'.

Les valeurs pour chacune de ces phases se donnent par l'instruction ENV. Mais comment faire?

Pensez-y et essayez!

Un bruit de tir atteindra très vite son volume maximum, retombera ensuite et résonnera.
La grosse caisse d'une batterie par contre se présentera comme un son pointu, en triangle; le battoir heurtera la peau et l'empèchera de vibrer ensuite.

l'enveloppe de tonalité

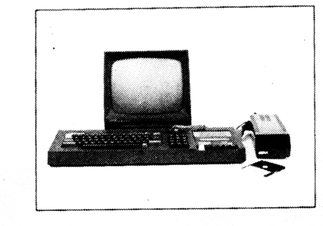
Son utilisation et sa syntaxe est similaire à l'enveloppe de volume.
Elle modifie légèrement la fréquence d'un son, en créant un effet vibrato.

Ici aussi, essayez, c'est la meilleure méthode car nous ne pouvons donner de règle précise, il n'y en a pas.


AMSTRAD NOUS VOILA !
MICRO APPLICATION
6
AMSTRAD
LA BIBLE
DU PROGRAMMEUR
DE L'AMSTRAD CPC
2
3
4
5
1
7

NUMÉRO 1 - MAI 85 - Prix 15 F. Belgique 120 FB. Suisse 4 FS.

MICRO INFO



LES PIRATES, CES GÉNIES MÉCONNUS !

LES PEEKS ET LES POKES DU COMMODORE 64

LE COMMODORE PC

DOSSIER : TOUT, TOUT, TOUT SUR LE COMMODORE 128

EXCLUSIF ! L'ATARI ST n'est plus qu'à 800 KM.

CONCOURS AMSTRAD
50 PRIX À GAGNER
Le programme le plus drôle !

Nous aimerions que l'information circule dans les deux sens !
Cette page vous est réservée.

NOM :
Prénom :
Adresse :

Code Postal :
Age :
Sexe :

Matériel utilisé :
Date d'Achat :
Extension/Périphérique :

Logiciel préféré :

Etes-vous satisfait des logiciels existant ? Oui Non

Si oui, lesquels ?

Si non, quels sont les logiciels que vous aimeriez trouver ?

Que pensez-vous des logiciels MICRO APPLICATION ?

Que pensez-vous des livres MICRO APPLICATION ?

Que pensez-vous de la revue ?

Votre rubrique personnelle :

ABONNEMENT

Attention Micro Info n'étant tiré qu'à 10 000 exemplaires.
Réservez dès à présent le numéro spécial Rentrée 85 plus les 3 prochains numéros pour la somme de 60 FF.

Règlement par chèque bancaire ou CCP uniquement.

Bulletin d'abonnement :

NOM :
Prénom :
Adresse :

Code Postal : Ville :

Nos petites Annonces gratuites seront réservées en priorité aux abonnés.

Achevé d'imprimer en juin 1985
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : juin 1985
N° d'imprimeur : 506001

IMPRIME EN FRANCE

[redacted]

L'AMSTRAD CPC dispose de capacités graphiques et sonores exceptionnelles. Ce livre en montre l'utilisation à l'aide de nombreux programmes intéressants et utilitaires.

Contenu :

- base de programmation graphique
- éditeur de police de caractères
- "sprites", "shapes" et chaînes
- représentations multi-couleurs
- calcul des coordonnées
- rotations, mouvements
- représentations graphiques de fonctions en 3D
- D.A.O. (dessin assisté par ordinateur)
- synthétiseur
- mini-orgue
- enveloppes de son

et beaucoup d'autres choses...

Ce livre est écrit par Jorg WALKOWIAK, étudiant en informatique et auteur talentueux de livres spécialisés (jeux d'aventures).

ISBN : 2-86899-016-9
129 F TTC

Ref. ML 124


AMSTRAD
GRAPHISMES ET SONS
DU CPC 464
8



Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/